AFRL-RI-RS-TR-2009-257
Final Technical Report
November 2009

OPEN COMPONENT PORTABILITY INFRASTRUCTURE (OPENCPI)

Mercury Federal Systems, Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* NOVEMBER 2009	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* January 2009 – July 2009

4. TITLE AND SUBTITLE OPEN COMPONENT PORTABILITY INFRASTRUCTURE (OPENCPI)	**5a. CONTRACT NUMBER** FA8750-09-C-0030
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 63781D
6. AUTHOR(S) John M. Scott, III	**5d. PROJECT NUMBER** SSTT
	5e. TASK NUMBER HG
	5f. WORK UNIT NUMBER 09
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Mercury Federal Systems, Inc. 1901 South Bell Street, Suite 402 Arlington, VA 22202-4511	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RITA 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** N/A
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2009-257

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-4668 Date Cleared: 12-Nov-09

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report presents the steps taken to research and convert the previously closed source software Component Portability Infrastructure (CPI) into Open Source Software (OSS) by releasing OpenCPI under an open source software copyright license agreement. By making the software OSS, the technology is suitable for a wider community, in part because the application and platform suitability has been previously verified through studies and reference applications. This report details research into: Intellectual Property (IP) issues and which OSS license to use, International Traffic in Arms Regulations (ITAR), classification issues, and the OSS business model for CPI. The intent and end goal is an open technology stack free of IP encumbrances and proprietary technologies. As a result of this research, CPI has been converted into OpenCPI and released under the OSS Lesser-GPL (General Public License), which is also called the GNU Lesser General Public License.

15. SUBJECT TERMS
Open Source Software, real-time, LGPL, military, copyright, intellectual property, FPGA, DSP, GPU, GPGPU, code portability, Component Portability Infrastructure, CPI, Open Component Portability Infrastructure, OpenCPI

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** George O. Ramseyer
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	48	**19b. TELEPHONE NUMBER (*Include area code*)** N/A

TABLE OF CONTENTS

1.0 SUMMARY 1

2.0 INTRODUCTION 3

2.1 WHAT IS CPI? 3
2.2 OPENCPI DESCRIPTION 4

3.0 METHODS, ASSUMPTIONS & PROCEDURES 8

3.1 OPENCPI RESEARCH AGENDA 8
3.2 OSS COMMUNITY RESEARCH AGENDA 9

4.0 RESULTS & DISCUSSION 10

4.1 GOVERNANCE & OSS LICENSING FOR OPENCPI 10
4.1.1 OPEN SOURCE SOFTWARE LICENSING 10
4.1.2 OSS LICENSING FOR OPENCPI 17
4.1.3 OSS GOVERNANCE 23
4.2 INTERNATIONAL TRAFFIC IN ARMS REGULATIONS (ITAR) 23
4.3 CLASSIFIED ISSUES 25
4.4 BEST PRACTICES OF OTHER EXISTING OSS COMMUNITIES 25
4.4.1 FINDABILTY 26
4.4.2 OPENCPI COMMUNITY RESOURCES 26

5.0 CONCLUSIONS 29

5.1 OPENCPI BUSINESS MODEL 29
5.1.1 OTHER DELIVERABLES 29
5.2 LOWER BARRIERS TO ENTRY 29
5.3 ENGAGE THE ACADEMIC & RESEARCH COMMUNITY 30
5.4 ENGAGE DEFENSE & INDUSTRY 30
5.5 ADD FEATURES AND APPLICATIONS 30

6.0 REFERENCES 31

7.0 LIST OF ABBRVIATIONS AND ACRONYMS 32

APPENDIX A: ADDITIONAL REFERENCES 34

APPENDIX B: MERCURY CPI ITAR REPORT 36

APPENDIX C: CPI DATASHEET 40

APPENDIX D: OPENCPI HARDWARE BILL OF MATERIALS 42

LIST OF FIGURES

Figure 1: CPI Description 4
Figure 2: CPI Data Flow Model 5
Figure 3: OSS Licenses Interaction[8] 15
Figure 4: CPI Export Control Summary 24
Figure 5: OSS and Classified Settings 25
Figure 6: OpenCPI Logo 26
Figure 7: OpenCPI Website 27
Figure 8: Example of OpenCPI Wiki 27
Figure 9: OpenCPI Software Source Code Website 28

LIST OF TABLES

Table 1: Software Licenses Combinations[9] 17

1.0 SUMMARY

This report details the steps taken by Mercury Federal Systems, Inc. (MercFed, website: www.mercfed.com) of converting its previously closed source software Component Portability Infrastructure (CPI) into Open Source Software (OSS), and its release under an open source software copyright license agreement. The application and platform suitability of this technology was verified through these studies and reference applications, and has resulted in the conclusion that it is suitable for a wider community. This effort focused upon the types of OSS licenses available, International Traffic in Arms Regulations (ITAR), classification issues, and the OSS business model for CPI.

Defense software is an instantiation of knowledge about a military capability, and this requires safeguards. However, hiding this knowledge behind Intellectual Property (IP) and other restrictions inadvertently ensures that the Department of Defense (DoD) and Air Force are less likely to reuse this existing defense software. This results in similar software being redeveloped and tested at government expense since the existing software is either not known to exist, or is protected by stringent IP or other restrictions. Additionally, this impairs the rapid development of new capabilities in an agile fashion.

IP rights are rarely being leveraged for the military's specialized needs. Since both the software and the hardware that the software is executed on are evolving so rapidly, the knowledge of defense software needs to be shared as widely as possible within the responsible community of military, industry and academia. However, at the same time, proper restrictions are required for that responsible community. The lack of an easily understood framework around exercising software IP in acquisitions leads to rising costs, slower innovation, and less agility.

Open Component Portability Infrastructure (OpenCPI) is used in the development of real-time signal processing for embedded, heterogeneous systems for communications, as for example xxxINT (Signals, Communications, Electronics, etc. - Intelligence) and Counter Improvised Explosive Devices (CIED) in defense Intelligence, Surveillance and Reconnaissance (ISR) systems. The intent and end goal of this effort is to have an open software technology free of IP encumbrance and proprietary uncertainty. The result of this research presented here is that the best pathway has been identified for CPI to be converted into OpenCPI and released under the OSS Lesser-GPL (General Public License, also called the GNU Lesser General Public License) to the responsible defense community.

Previously CPI was targeted and validated in selected domains, but its applicability across a wider scope of multi-processing technologies and application areas has only been asserted based on extrapolation. Furthermore, the degree of openness for a wide array of platforms, supportable by various organizations (proprietary contractors, government programs, commercial-off-the-shelf (COTS) providers), has only to be "designed in", and not "validated in practice".

This effort targeted high priority candidates in three categories for prototyping and assessment. A web based community framework was populated, and the technical and community issues occurred by CPI as it was transitioned to open source software (OpenCPI) were documented. Issues that were addressed included design artifacts, testing suites, software developer kit, IP regime and repository issues, code governance, business model and community design. Building the community around an open source software project will take time, so here we focused on gathering like-minded individuals who have the problem sets of real-time and embedded software.

One of the best ways to grow and maintain a healthy software baseline is to ensure there are revenue opportunities available. The most successful company that has been able to monetize open source software is RedHat with its two main product lines: RedHat Linux and JBoss. RedHat provides a subscription service to both of these products that includes help line support, rapid updates of bug and new features, and tool sets to help manage the product lines internal to a customer. Sun Microsystems also sells a subscription support service for all of its associated open source software products such as OpenSolaris and Java. There are a number of different business models, listed here in order of lower to higher value: Selling installation with service and support with the software, versioning the software, free versions as an entry-level offering and other, more advanced versions as value-added offerings, integrating the software with other parts of the customers information technology infrastructure, and lastly providing proprietary complements to open source software.

To broaden the community, the barrier to entry to using OpenCPI must be lowered. This means simplifying the installation and use process of OpenCPI. A hardware bill of materials was created for download on www.OpenCPI.org to illustrate a simplified example of what an OpenCPI system would look like. Developer and build notes about the sample systems were posted. OpenCPI needs to have multiple communication channels, including a blog, wiki, email lists, a twitter feed and eventually an always on IRC (internet chat relay) chat channel so that developers and users feel they always have 24/7 distributed support.

Most open source software projects have seen great adoption by users in the academic and research communities, since the technologies are so accessible to use and extend. OpenCPI has reached out to the academic community and is slowly gaining interest and traction. A key future feature would be to fund academic research and development projects to show how OpenCPI can be a key technology enabler. The community that will have the most to gain from OpenCPI across a number of application areas is the military, so it is vital that MercFed engages with as many current and potential military users of OpenCPI as possible.

2.0 INTRODUCTION

It is the intent of this report to provide guidance on how to convert closed and proprietary software into open source software, and is focused primarily on those tools needed by the US military. This report is provided for information only, and while summarizing legal advice and guidance, no one who contributed to this report is a lawyer. Please consult proper legal counsel as required when releasing software source code.

Section 2 is the Introduction, which describes OpenCPI. Section 3, Methods, Assumptions & Procedures, describes the process we went through to complete this research. Section 4, Results & Discussion, presents the results of this research and details potential issues to be avoided when converting software to open source. Section 5, Conclusions, lists conclusions of this research. The appendices (A-D) provide additional detail on references used, software descriptions, an export control review and hardware bill of materials, respectively.

2.1 What is CPI?

The Component Portability Infrastructure (CPI) is an innovative middleware solution that simplifies the programming of heterogeneous processing environments consisting of field-programmable gate arrays (FPGAs), general-purpose processors (GPPs), digital signal processors (DSPs), and high-speed switch fabrics. OpenCPI greatly improves code portability, interoperability, and performance in FPGA and DSP-based environments by providing well-defined waveform component APIs with a set of infrastructure building blocks that act as a hardware abstraction layer (HAL).

Today's myriad communications standards and rapidly evolving new-generation waveforms have created a need to build communications systems that are ready to accept any present or future waveform. Waveform-Ready™ processing platforms combine the latest processor, transceiver, and interconnect technologies with the CPI to help customers meet this challenge. Building on the concepts introduced by the U.S. Government’s Software Communications Architecture (SCA) standard, CPI extends component-based architectures into FPGAs and DSPs to decrease development costs and time to market through code portability, reuse, and ease of integration. CPI has over 30 man-years of development and is considered a military Technology Readiness Level (TRL) of 6.[1]

CPI is used in real-time signal processing for embedded, heterogeneous systems for communications, xxxINT (Signals, Communications, Electronics, etc. - Intelligence) and CIED (Counter Improvised Explosive Devices) in defense intelligence, surveillance and reconnaissance (ISR) systems. While CPI has been targeted and validated in certain domains, its applicability across a wider scope of multi-processing technologies and application areas has only been asserted based on extrapolation. Furthermore, the degree of openness for a wide array of platforms, supportable by various organizations (proprietary contractors, government programs, COTS providers) has only to be “designed in”, and not “validated in practice.”

2.2 OpenCPI Description

CPI is a real-time embedded (RTE) middleware solution that simplifies the programming of heterogeneous processing applications requiring a mix of field-programmable gate arrays (FPGAs), general-purpose processors (GPPs), digital signal processors (DSPs), and high-speed switch fabrics. The “mix” can be over a lifecycle (technology insertion), or within a single implementation. CPI improves code portability, interoperability, and performance in FPGA and DSP-based environments by providing well-defined waveform component Application Programming Interfaces (APIs) with a set of infrastructure blocks that act as a Hardware Abstraction Layer (HAL). CPI is also appropriate for the incorporation of GPU and multicore technologies. CPI is uniquely positioned to meet the goals of the Software Systems Stockroom (S3) since in some sense component-based systems are computer-science's answer to dealing with “knowledge capture" and the lock-up of Intellectual Property (IP). All interfaces are openly published and non-proprietary, using an appropriate mix of industry and government specifications.

To overcome the challenges of code portability in FPGA environments, CPI provides a pre-validated set of building blocks (Figure 1) to interface the FPGA waveform applications with high-performance switch fabrics, onboard memory, system command and control, and wideband Input/Output (I/O). CPI's non-proprietary interfaces act as an abstraction layer to increase the portability of FPGA applications. A verification suite is also included to facilitate debugging and reduce development time.

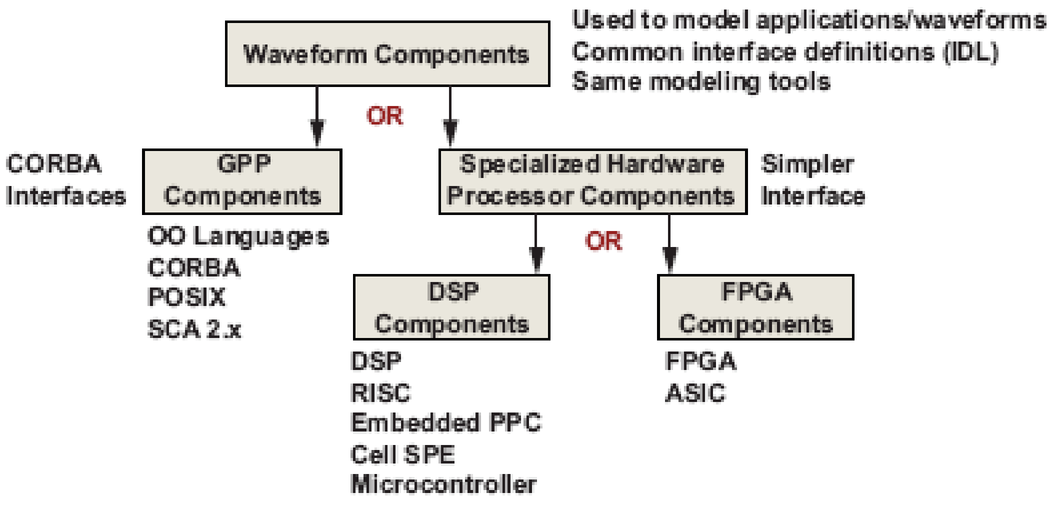


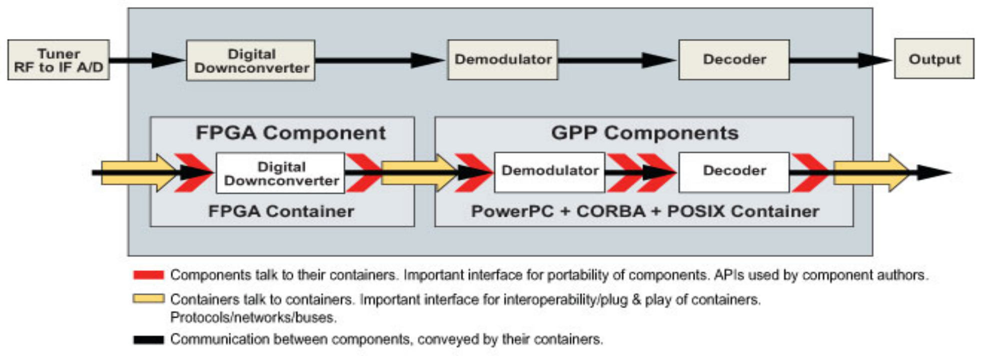
Figure 1: CPI Description

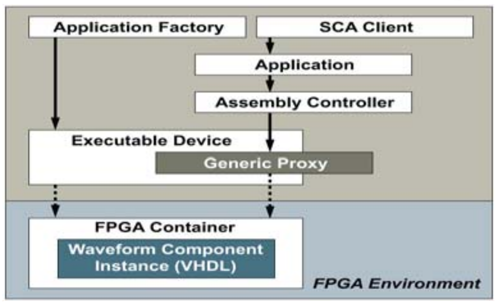
At the highest level, the CPI vision allows users to outsource the technology transition management job to others. Using the CPI interfaces, developers can protect their application development investment by cost-effectively moving their applications to new generations of systems using the latest technologies. CPI is essentially a kit of necessary pieces to create an application platform for component-based applications based on the SCA model extended to a heterogeneous mix of computing, interconnect and I/O resources. When CPI is adapted to, and installed on a platform, that platform is said to be “waveform-ready”.

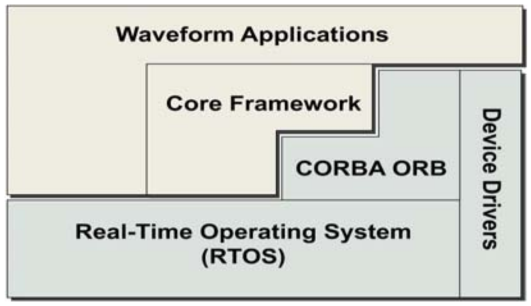
While the SCA defines the operating environment and APIs for C++ software applications components running in the Common Object Request Broker Architecture (CORBA) and Portable Operating System Interface [for Unix]**-** **(**POSIX-)compliant environment, CPI extends the SCA environment, according to the Proposal 289 to SCA (FPGA/DSP extension by Mercury funded by program office) to DSP and FPGA technologies. Analyses for suitability for Graphics Processing Unit (GPU) and Multicore technologies have shown promise.

Middleware for Waveform-Ready™ Processing Platforms

- **Improved waveform code portability with standards-based interfaces**
- **Increased interoperability using container technology**
- **Quicker time to market**


Tuner
RF to IF A/D
Digital
Downconverter
Demodulator
Decoder
Output
FPGA Component
Digital
Downconverter
FPGA Container
GPP Components
Demodulator
Decoder
PowerPC + CORBA + POSIX Container
Components talk to their containers. Important interface for portability of components. APIs used by component authors.
Containers talk to containers. Important interface for interoperability/plug & play of containers. Protocols/networks/buses.
Communication between components, conveyed by their containers.





FPGA example Using CPI

SCA is implemented as a client

Figure 2: CPI Data Flow Model

For FPGA environments CPI uses the industry-standard Open Core Protocol (OCP) to define language-independent interfaces (Figure 2). OCP delivers a non-proprietary, openly licensed, core-centric protocol that comprehensively describes the system-level integration requirements of IP cores. OCP eliminates the task of repeatedly defining, verifying, documenting, and supporting proprietary interface protocols.

A clear advantage of using OCP to describe a core's interfaces is that the mechanisms through which one OCP interface can talk to another are clearly defined by the OCP specification. Even if two connected cores have dissimilar interfaces, the fact that they are valid OCP interfaces means that the information needed to resolve those dissimilarities is readily available. CPI builds on well-defined OCP-compliant profiles to define signals and semantics for control, configuration, data, and memory interface patterns. These OCP profiles support waveform component control and configuration, First In - First Out (FIFO) streaming with flow control, message passing with random addressing and buffer reuse, and memory interfaces for Static Random Access Memory (SRAM) or Dynamic Random Access Memory (DRAM) as well as on-chip memories.

The dominant view of CPI is as an application framework that makes many complex underlying platform issues go away, and extends the useful lifetime of application code in the face of technology transitions. All parts of CPI are either focused on directly providing the application environment, or providing plumbing, middleware, drivers, and even development tools to support the application model.

Applications of CPI include communication terminals (including high rate satellite communications), counter-IED (Improvised Explosive Device) and Signals Intelligence (SIGINT) equipment, packet inspection, and other data exploitation missions. The list is limited primarily by those applications eager or required to exploit mixed technologies, but clustered or co-simulated systems are also appropriate. Modern communications systems are designed to support and switch between multiple "waveform" applications. In certain cases, the systems are designed without even knowing the application that will eventually run on them. This new approach requires a departure from traditional design methodologies as well as what the design engineers need in a development environment.

CPI allows users to create integrated heterogeneous subsystems, called Waveform-Ready™ platforms, and enables users to focus on their application without worrying about platform-specific details such as interconnect technologies, memory access, or data acquisition.

As Software Designed Radio (SDR) technology moves from narrowband radios to wideband data links and satellite communications, hardware designs rely more on FPGAs. However, FPGA developers are furthest away from the enabling technologies of SDR such as component-based programming and code reuse. This disconnect helps CPI offer the most value in FPGA-heavy systems today.

To summarize, CPI helps with:

a. ***Code Portability:*** CPI increases the portability of waveform applications in heterogeneous processing platforms by providing APIs, software modules, and intellectual property cores that abstract the complexities of the underlying hardware platform away from the application developer. This layer, known as containers, provides the control, configuration, and communication abstractions consistent with system-level component architectures compatible with the SCA. It exploits the underlying hardware capabilities and performance, while dramatically reducing dependencies of application code on platform technologies, topologies, and configurations. Waveform components use CPI's containers through open and non-proprietary interfaces. Because the interfaces are standardized, they can be reused in any other platform that supports the same interfaces.

b. ***Open Core Protocol (OCP) Profiles:*** CPI uses **the industry-standard** Open Core Protocol to define the interfaces for FPGA environments. CPI uses well-defined OCP-compliant profiles to define signals and semantics for control, configuration, data, and memory interface patterns. CPI will continue to build on SDR concepts, but will be extended to more demanding, more heterogeneous technologies such as General-Purpose computing on Graphics Processing Units (GPGPUs) and multi-core environments.

c. ***Increased Interoperability:*** CPI also facilitates the interoperability of components executing on different computing device technologies. Typical waveform applications consist of multiple distributed components operating within a heterogeneous embedded environment. CPI provides an environment for FPGA, GPP, and DSP components to interoperate seamlessly. When components communicate with one another, their containers mediate the communication and route it over the appropriate path.

d. ***Quicker Time to Market:*** CPI allows an application to be brought to market faster. CPI’s abstraction layer eliminates the need for the application developer to become intimately familiar with the underlying hardware platform. The developer needs to understand and work with only the open and non-proprietary component interfaces that are supported by CPI. In addition, CPI’s standard interfaces facilitate the reuse of IP, thereby allowing developers to integrate existing components onto the platform quickly and easily. “Time to market” includes support of rapid technology insertion when parts of applications are retargeted at very different technologies as they become available.

e. ***Subsystem Integration*****:** CPI enables users to create an integrated subsystem “out-of-the-box” saving money and more importantly, saving time.

3.0 METHODS, ASSUMPTIONS & PROCEDURES

There is considerable latent value in the defense software domain owned by defense companies. Here we clearly document how to transition software code from a proprietary and closed environment into an OSS business model. This transition is heavily documented so that in the future should the government (or a contractor) wish to make open source software code, this roadmap of the transition process will be available. This benefits the wider community by making the technology suitable and available, and additionally the application and its platform have previously been verified through studies and reference applications.

The first OpenCPI research item addressed was which type of OSS license to use. Different licenses engender different types of group behaviors. For instance, a Berkeley Software Distribution (BSD) OSS license allows users to compile and sell OSS BSD code as long as the copyright notice is attached with the code. By contrast, the GNU General Purpose License (GPL) is the most restrictive in terms of changes made to the source code. Before changes can be distributed, it is required that changes be shared back to that software community.

Another issue examined is how to mix software that is open source, unclassified and classified. This is a key to proving an OSS business model that can truly function in the defense market. Research included the examination of the OSS business model for CPI. The requirements for this transaction were within the scope of this project. The OSS model is intended to aid in the adoption of CPI throughout the defense community. ITAR issues were also examined.

3.1 OpenCPI Research Agenda

The OSS community envisioned here is built around OpenCPI, and is focused entirely on the needs of the military. Military industry partners were brought together to focus on the problems of code portability, while being hardware agnostic for Real-Time Operating System (RTOS). Team members had previous experience with CPI, and were very interested in seeing it become open source. In fact, one of the team members commented that if CPI became OSS it would change how RTE systems are built, deployed and supported.

The approach of this effort was the development of a prototype community driven website where the users and developers of CPI interact and negotiate to add features and mature the CPI code-base. The community helps mediate discussions around CPI needed capabilities, new ideas, bug fixes, additional features and find support for problems and issues that occur with CPI. A standard open source web community was utilized for this. The Software Systems Stockroom (S3) community environment adapted the following tools to capture knowledge about CPI: OSS Eclipse Modeling & Development Framework and the OSS Trac Integrated Software Project Management (includes collaborative suite of software tools: email lists, wiki, subversion code framework, etc.).

One key issue that is sometimes overlooked in code is that occasionally the key piece of information is not listed in the design documents or the code comments, but is sometimes contained in emails of online chat sessions. The community S3 that was built was focused on how to harvest, store and ensure easy "findability" of unstructured content around CPI capabilities.

Defining a governance structure was also important. A good governing structure in a software community ensures that the community will cohesively survive versus fracturing and dying.

The key to developing and evolving a robust community that meets the needs of the members and the government is the option to take the code and leave. The CPI community focuses not just on CPI source code, but also on a lifecycle approach for how to build applications on top of CPI. Test suites will be built on for how to test CPI components.

There were ultimately three research areas: 1) research how-to convert an RTE military technology into open source software, 2) research, development and refinement of the CPI platform and 3) develop an effective community to share knowledge and ultimately drive development to meet new threats and new capabilities.

The core research agenda was to “open up” this component-oriented framework and architecture consisting of experiments to verify (or disprove) this applicability in three dimensions:

1. Processor and interconnect technology agility and supportability
2. Platform/system supportability
3. Application domain suitability (ease-of-use, natural implementation models etc.)

3.2 OSS Community Research Agenda

Questions to be answered and research deliverables for this report included:

1. What OSS license should be used?
2. How should the community be governed, e.g., How should voting occur on the CPI codebase?
3. Define a business model for OpenCPI.
4. Examine ITAR and classification issues with CPI.
5. Define how to scale OpenCPI community in Phase 2.
6. Deliverable: CPI conversion to OSS in Phase 1 with associated collaboration website.
7. Deliverables: OSS roadmap conversion for the defense industry and OpenCPI website user and administration guide.

4.0 RESULTS & DISCUSSION

This section describes our results of converting CPI to OpenCPI and details the associated steps of coming to a decision of which OSS license to use, how to govern an OSS community focused around the military, releasability issues (if any) and ultimately how to scale and grow the envisioned OpenCPI community.

4.1 Governance & OSS licensing for OpenCPI

As part of the Air Force Research Lab (AFRL) Software Systems Stockroom (S3) BAA research program, Mercury Federal Systems, Inc. (MercFed) proposed to convert its Component Portability Infrastructure (CPI) into open source software (OSS). To accomplish this, research had to be done to answer the following key questions:

1. What OSS license should CPI be released under?
2. Are there any export restrictions associated with CPI?
3. What should the CPI community look like and what tools will they need?
4. How does MercFed go about this process to help to inform and educate other military focused companies to do the same? How can this process be documented?

4.1.1 Open Source Software Licensing

Why attach an open source copyright license to software source code? The simplest reason to attach an OSS license to software is to help users, individuals, and organizations easily use and modify software. The argument is this: if there is not a license attached to a piece of software, and the software is in the public domain (or on a website), how can anyone know what the acceptable use of the software? Who wrote it? Where did the intellectual property come from? Who owns the original copyright and trademark? Most importantly, what is an individual allowed to do with that software?

An OSS license structures an answer to these questions for a potential user of a software program. An OSS license lays out what rights and responsibilities are incumbent upon a user of a software program. Developers might like ongoing recognition of their efforts or control of the original trademark. They may also prefer that any changes to the source code be passed along to any downstream user of that software.

First, what defines an "open source software license? Choosing an OSS license can be daunting; at last count there were over sixty licenses[2] deemed to comply with the open source definition.[3] Open source doesn't just mean access to the source code. The distribution terms of open-source software must be accompanied with an OSS license that states what the terms and conditions are that the users of the software need to comply with.

The non-profit Open Source Initiative[4] (OSI) reviews potential licenses and judges their worthiness to be called open source. The ten tenants are:

1. Free Redistribution: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code: The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works: The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code: the license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups: The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor: The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product: The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software: The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral: No provision of the license may be predicated on any individual technology or style of interface.

A full listing of the licenses can be found on the OSI website at http://www.opensource.org/licenses/alphabetical. A license that incorporates each of these three freedoms also includes provisions related to the intertwined issues of copyright, intellectual property and trademark.

4.1.1.1 Copyright & License

Currently, copyright law applies the moment an original work is created. In the case of CPI, the original work was authored by the Mercury Computer Systems, Inc. Mercury owns the entire copyright to the source code in CPI and thus can do anything it wishes with it. This includes giving CPI away for free, destroying/deleting it, or licensing it for sale. MercFed decided it was in the company's best interests to convert CPI into an OSS licensed product and offer it to customers under a subscription service.

Copyright laws govern OSS licenses. A copyright provides exclusive rights to do certain things with the intellectual property that others cannot do without your permission.[5] Rights that primarily concern software are:

- ➢ An exclusive right to make copies.
- ➢ An exclusive right to prepare derivative works.
- ➢ An exclusive right to distribute copies of the original or derived works.
- ➢ The full list of rights can be found in the U.S. Copyright Act, 17 U.S.C. §106.

In essence, copyright allows the owner exclusive rights to do certain things. The copyright owner can then grant a license to someone else to copy, modify, or distribute a piece of software. A license gives boundaries around the use of a piece of Intellectual Property (IP) and provides conditions on its use. Licenses also allow IP owners recourse if conditions are not met or boundaries are violated in the course of using the IP. In the case of OSS, licenses can be considered a contract, although there are differing views on this.

4.1.1.2 Intellectual Property & Patents

Patents dictate and enforce how an original idea is controlled. Patents give an owner the *right to exclude others from doing* certain things with patented intellectual property:[6]

- ➢ The right to exclude others from making products embodying your patented invention.
- ➢ The right to exclude others from using products embodying your invention.
- ➢ The right to exclude others from selling or offering for sale products embodying your invention.
- ➢ The right to exclude others from importing products embodying your patented invention.
- ➢ The full list of rights can be found in the U.S. Patent Act, 35 U.S.C. §154.

A patent grant is an affirmative license to practice patents necessary to make, sell or offer for sale, or import the software, but only to the extent of patent claims actually owned or controlled by the licensor.

Since software sometimes had patents associated with its use, some of the OSS licenses provide a license to users of that software to use and extend the software without conditions except those embodied in the license.

4.1.1.3 Trademark

Trademark provides exclusivity over the use of a name. For example: RedHat Linux is a branded version of Fedora Linux, and only RedHat can market and sell the RedHat version of Linux. This allows Redhat to create a brand around their version of Linux. Most successful open source software projects claim trademark over the name of an OSS project to ensure that quality can be met and maintained.

Registering a trademark can be either a relatively easy process or an expensive process, depending on how much protection a company desires. Details on how to register a trademark can be found on the U.S. Patent and Trademark Office's website: http://www.uspto.gov.

4.1.1.4 OSS License Types

The over sixty types of OSS licenses can be grouped into three categories: permissive (or Academic), partially closable, and reciprocal. Permissive licenses are by far the most open, and allow the software source code to be used in any fashion by a user. Permissive licenses are also referred to as academic licenses, since a majority of them came out of an academic environment, where making ideas accessible is of paramount importance.

The three general classes of software licenses are:

a. Permissive (or Academic)

- ➢ Allows unfettered access to the code.
- ➢ Can be added into proprietary applications and resold or relicensed.
- ➢ There are no requirements for downstream sharing of source code.
- ➢ Examples of permissive OSS licenses include BSD, MIT, and Apache.

b. Partially Closable

- Proprietary applications can use an unmodified version of a software library in a closed source, proprietary licensed product. For example, if changes are made to a software library under the Lesser General Public License (LGPL), the modified source code (corresponding to that library) along with the binary application must then be distributed to the end users.
- LGPL projects include Jboss and Open Source Software Image Mapping (OSSIM)

c. Reciprocal

- Requires licensee of the code to reciprocally apply the same open source license to any code derived from the originally licensed code.
- Each binary distribution also includes the application's full source code.
- Reciprocal licenses include the General Public License (GPL).
- The best known GPL example project is Linux.

OSS licenses, like software and other technologies, have adoption lifecycles that show up in the statistics of what licenses are the most widely used. Statistics from a study[7] presented on the OSS software hosting website Freshmeat.net (on 11/10/2003) show that almost 70% of the 32,592 OSS projects surveyed were GPL-licensed, while LGPL and BSD were 5.2% and 4.8 %, respectively. The other major OSS project hosting website SourceForge.net (on 11/10/2003) showed that GPL projects accounted for 71% of 45,736 projects hosted, with LGPL at 10% and BSD at 7%. While neither was an exhausting survey, it showed that GPL and GPL-compatible licenses predominantly share the OSS license market.

4.1.1.5 OSS License Combinations

Simply picking and using an OSS license wasn't the end of the process. Software is usually combined and recombined with other types of closed and open source software, resulting in software that is governed by one or both of the licenses. Great care should be taken when picking a license to ensure that a software program can be used and reutilized by the widest group of users as possible.

As illustrated in Figure 3, only some OSS licenses can be combined with other types of licenses while meeting the requirements of those licenses.

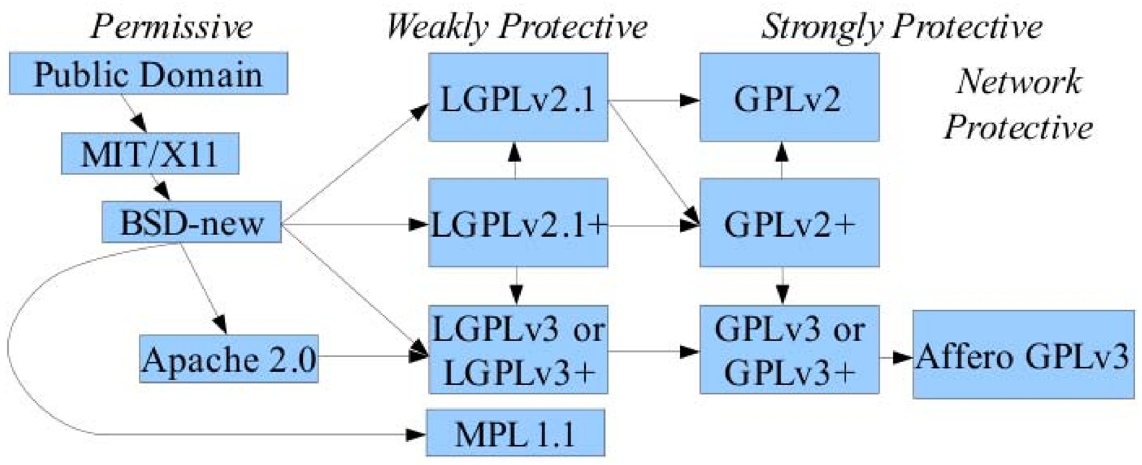


Figure 3: OSS Licenses Interaction[8]

To quote David Wheeler[8] at length:

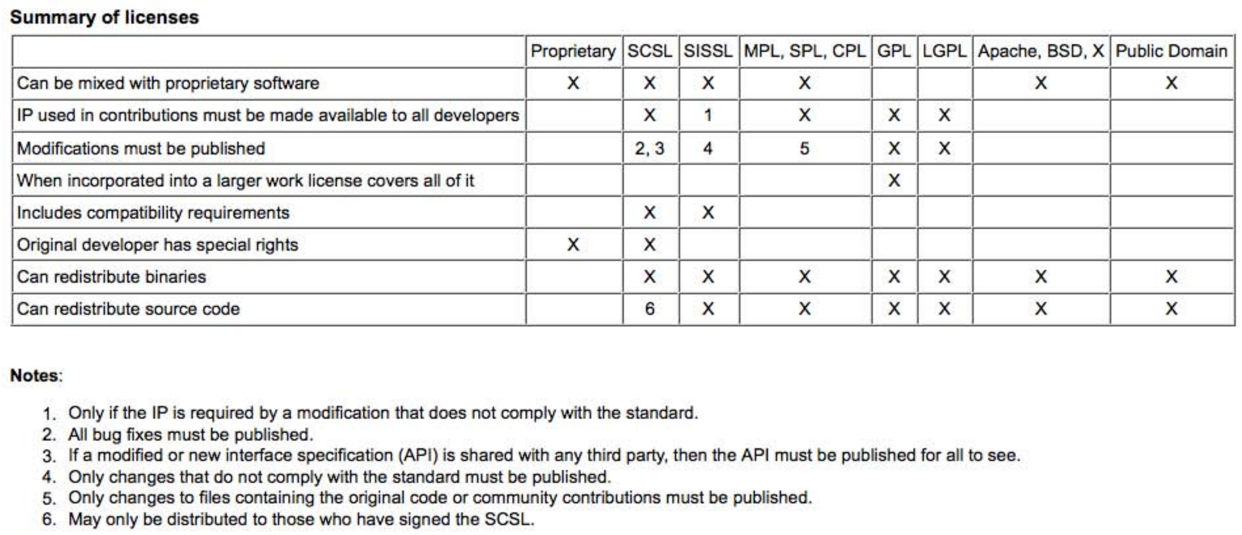
> “In this figure [Figure 3], the shaded boxes are the names of different FLOSS licenses. An arrow from box A to box B means that you can combine software with these licenses; the combined result effectively has the license of B, possibly with additions from A. To see if software can be combined, just start at their respective licenses, and find a common box you can reach following the arrows (aka “following the slide”). For example, Apache 2.0-licensed software and GPLv2+-licensed software can both reach “GPLv3 or GPLv3+”, so they can be combined using GPLv3 or GPLv3+. This figure has been carefully crafted so following a path determines if two licenses are compatible. For more information you must examine the license text, but this gives the basic answer quickly.
>
> At the left are the “permissive” licenses, which permit the software to become proprietary (i.e., not FLOSS). At the top left is “Public Domain”, which strictly speaking isn’t a license but in effect it works like one. You can do anything with public domain software, but it is rare; the software must be explicitly released to the public domain or be created by a U.S. Government employee in their official capacity. Next is the so-called “MIT” or “X11” license, which is very permissive (you can do just about anything except sue the author). Software under the MIT license is easily combined with the modern 3-clause Berkeley Software Distribution (BSD-new) license, which compared to the MIT license adds a clause forbidding the use of the author’s name to endorse or promote products without permission (it’s debatable if this clause actually does anything, since you typically have to have such permission anyway). Finally we have the Apache version 2.0 licenses.

At the right are the “strongly protective” (“strong copyleft”) licenses, which prevent the software from becoming proprietary. This includes the most popular FLOSS license, the GNU General Public License (GPL). The GPL has a version 2 (GPLv2) and 3 (GPLv3); a “+” afterwards means “version X or later”. GPLv2-only cannot be combined with the network-protective Affero GPLv3, but GPLv2+ (“version 2 or later”) can via GPLv3.

In the middle are the “weakly protective” (“weak copyleft”) licenses, a compromise between permissive and strongly protective licenses. These prevent the software component (often a software library) from becoming proprietary, yet permit it to be part of a larger proprietary program. This figure shows the rules when you are making other software part of the weakly protected component; there are other possibilities if you are only using the component as a library. The GNU Lesser General Public License (LGPL) is the most popular weakly protective license, and has a version 2.1 (LGPLv2.1) and 3 (LGPLv3). Note that LGPLv2.1 gives you permission to relicense the code under any version of the GPL since GPLv2. Another such license is the Mozilla Public License 1.1 (MPL 1.1), but the MPL has the serious drawback of being incompatible with the widely-popular GPL; you can’t even use an MPL module in a larger GPL’ed program.”

In Table 1 the website Java.net has a more detailed version of how to combine software licenses and the software freedoms espoused by Open Source Initiative (OSI) and then how that resulting software can be shared.

Table 1: Software Licenses Combinations[9]



Summary of licenses

	Proprietary	SCSL	SISSL	MPL, SPL, CPL	GPL	LGPL	Apache, BSD, X	Public Domain
Can be mixed with proprietary software	X	X	X	X			X	X
IP used in contributions must be made available to all developers		X	1	X	X	X		
Modifications must be published		2, 3	4	5	X	X		
When incorporated into a larger work license covers all of it					X			
Includes compatibility requirements		X	X					
Original developer has special rights	X	X						
Can redistribute binaries		X	X	X	X	X	X	X
Can redistribute source code		6	X	X	X	X	X	X

Notes:

1. Only if the IP is required by a modification that does not comply with the standard.
2. All bug fixes must be published.
3. If a modified or new interface specification (API) is shared with any third party, then the API must be published for all to see.
4. Only changes that do not comply with the standard must be published.
5. Only changes to files containing the original code or community contributions must be published.
6. May only be distributed to those who have signed the SCSL.

4.1.2 OSS Licensing for OpenCPI

For CPI to be converted four assumptions were generated in the process of researching OSS use and conversion of closed source into OSS in a real-time, embedded software environment.

Assumptions, with justification, are:

- ➢ Must be GPL compliant, to enable maximum reuse of CPI with other OSS programs

GPL license compliance is key as noted previously. GPL and GPL-variant licenses overwhelming are the choice for the wider OSS community. Together GPL-type licenses control near 80% of the market. Picking a license that can't be combined with other software projects would limit the ability of OpenCPI to be used by the wider defense community, in effect creating an "OpenCPI" island where only CPI-related code can be comingled.

- ➢ Do not create new or 'special' Government or Company specific OSS license

There is a history of organizations and companies concluding that they want to release software as open source, but they also desire additional rights. This has lead to license inflation, with too many licenses to choose from, and even worse, software packages being unable to inter-operate with respect to software license because no one has completed the legal analysis to decide whether (to assumption #1) they could be combined, for example with the GPL or Apache OSS licenses. In general: the rule in the OSS community is, don't create new OSS licenses.

- ➢ The CPI OSS license needs to 'fit' into the real-time embedded software community

MercFed needs to be cognizant that OpenCPI is meant to be used by the real-time embedded software community, and needs to understand how this community views OSS systems and integrates software into its mission sets.

In general, the real-time embedded market is focused on the high-speed performance of systems. For example, these systems are very tightly coupled with software that requires real-time systems using GPL licensed and modified code to be shared back with the GPL. OpenCPI is a middleware product connecting hardware compute resources with applications. For example, the intellectual property in a FPGA is sometimes proprietary, as can be the application suite of tools. Real-time embedded systems can also be heavily modified and statically linked, which requires source code to be shared under the GPL. For this reason using the GPL license is not a viable option at this time for OpenCPI, since vendors and systems integrators who would use OpenCPI would tend to shy away from GPL encumbered licenses. For this very reason some defense contractors also have prohibitions against using GPL licenses outside of operating systems.

- ➢ A CPI OSS license needs to lower the friction of using OpenCPI and foster the greatest use possible, leading to a community that grows and scales to fit the needs of the military.

To follow these points, any real or perceived barriers to use OpenCPI in the defense industry must be eliminated. Choosing a license where there is a prohibition may affect longer-term growth and potential for CPI.

4.1.2.1 OpenCPI Licensing

After completing the research of open source software licenses (and due to Section 4.1.1.4 – OSS License Types), it was determined that there are only 5 license types to consider (note: all licenses are versioned):

- ➢ Apache
- ➢ Mozilla Public License (MPL)
- ➢ Berkeley Software Distribution (BSD) / MIT variants
- ➢ Lesser General Public License (LGPL)
- ➢ GNU General Public License (GPL)

Most of these licenses have several versions (and per Section 4.1.1.5 – OSS License Combinations), and some cannot be combined together. For example, according to Free Software Foundation & others[10], the Apache license version 1 – 1.1 and the MPL are not GPL compliant

4.1.2.2 Context of OSS Licensing for OpenCPI

Our plan is for OpenCPI to be used extensively within the US military and associated defense markets in the real-time and embedded software domain. The license chosen for OpenCPI also needs to be compatible with other OSS projects in the military embedded space as well.

A few key OSS projects used in the military and embedded software space include:

- RedHat (http://www.redhat.com/) is the premier Linux and middleware vendor to the US military. They utilize the GPL v3.0 and a commercial license to support Linux, and the LGPL v 2.1 and a commercial license for the middleware JBoss support.

- NetFPGA (http://www.netfpga.org/) is an OSS project to enable researchers and students to build working prototypes of high-speed, hardware-accelerated networking systems. The NetFPGA is a line-rate, flexible, and open platform for research and classroom experimentation. About 1,000 NetFPGA systems have been deployed at over 120 institutions in over 15 countries around the world. The NetFPGA is used by many classroom teachers to help students learn how to build Gigabit Ethernet (GigE) switches and Internet Protocol (IP) routers. It has also been used by researchers to prototype new modules that use hardware rather than software to forward packets, and is governed by a BSD-style OSS license.

- VSIPL (http://www.vsipl.org/) stands for Vector Signal Image Processing Library, and is an application programming interface that is defined by an open standard. It was developed by embedded signal and image processing hardware and software vendors, academia, application developers, and government laboratories. A number of hardware and software vendors have VSIPL products, and it is being increasingly used by developers who desire a highly efficient and portable computational middleware for signal and image processing applications. It uses a BSD-style OSS license.

- RTLinuxFree (http://www.rtlinuxfree.com/) provides a mechanism for downloading, evaluating, and using RTLinux technology, subject to the terms and restrictions of the Open RTLinux Patent License. The license requires that all applications using the free RTLinux download be GPL licensed, and also requires that the application source code be made publicly available on the Web. In exchange, Wind River provides royalty-free use of the patented RTLinux process. Open RTLinux is for academic research and other open software projects that include real-time functionality on a self-supported, free software basis. For additional information, see the Open RTLinux Patent License. Wind River Real-Time Core is for projects with stringent requirements for off-the-shelf, guaranteed real-time functionality in commercial-grade software, accompanied by professional support and services to meet pressing time-to-market goals. RTLinuxFree.com is governed by both the GPL and a commercial license.

- OpenSAF (http://www.opensaf.org/) is an Open Source Project established to develop a base platform High Availability middleware consistent with Service Availability Forum™ (SA Forum) specifications under the LGPLv2.1 license. The OpenSAF Foundation was established by leading Communications and Computing Companies to facilitate the OpenSAF Project and to accelerate the adoption of the OpenSAF code base in commercial products. The objective of the new OpenSAF project is to accelerate broad adoption of an SA Forum compliant operating environment. The goals of the OpenSAF project are:

- Create an open source implementation of a high availability-operating environment, which includes the SA Forum Application Interface Specification.
- Develop the necessary additional complementary services required to deploy and manage software.
- Accelerate the development of SA Forum specifications by proposing enhancements implemented in the OpenSAF project.
- Establish a broadly adopted high availability operating environment that can be leveraged by computing technology companies, NEPs and other industries requiring high availability and ISVs.
- Utilize an open source licensing model not tied to any commercial implementation.

➢ Qt (http://www.qtsoftware.com/products/) is a cross-platform application and user interface framework. Using Qt, applications can be written once and then deployed across many desktop and embedded operating systems without rewriting the source code. Qt uses the LGPL v2.1 and GPL 3.0 OSS licenses and a commercial license.

➢ OpenEaagles (http://www.openeaagles.com/docs.html) is an open source framework designed to support the rapid construction of virtual (human-in-the-loop) and constructive simulation applications. It has been used extensively to build DIS compliant distributed simulation systems. Serving as a simulation design pattern it provides a structure for constructing simulation applications. The framework aids the design of robust, scalable, virtual, constructive, stand-alone, and distributed simulation applications. It leverages modern object-oriented software design principles while incorporating fundamental real-time system design techniques to meet human interaction requirements. It is released under the LGPL V 3.0 OSS license.

The defense community is unofficially resistant[1] to using the GPL license outside of Linux software programs. The main reason seems to be misconceptions on what GPL license requirements means with respect to publishing and distributing of source code. A number of vendors are reluctant to be in a position where they may have to distribute source code if modifications are made to an OSS GPL program.

[1] Not-for-attribution conversations with individuals in defense companies

There is a large amount of OSS software outside of the GPL used throughout the defense community. As a 2003 MITRE report[11] pointed out if DoD removed all OSS from its systems, it would simply cease to function as an enterprise. OSS is obviously used in classified settings. In fact a few projects listed below have figured out how to deploy and use OSS in unclassified and classified settings:

- Open Source Software Image Map (OSSIM) (http://www.ossim.org) provides advanced geo-spatial image processing for remote sensing, photogrammetry, and Geographic Information Systems. Backed by an active open source software development community, OSSIM solutions have been deployed on a number of critical commercial and government systems. OSSIM is used at military and intelligence agencies such as NGA and NRO in highly classified settings.

  - OSSIM classified code is kept on a JWICS (IC network) website for anyone with the IC to use. Bug-fixes and new ideas for features and upgrades are worked on in an unclassified setting, and published into the public OSS website if those fixes and features are of an unclassified nature.
  - New public OSS OSSIM source code is burned onto a CD and brought into a classified setting. No source code is ever brought out from a classified setting due to information security requirements.
  - OSSIM is supported by RadiantBlue, Inc., who insures the security of all source code before military use is authorized.

- Opticks (https://opticks.ballforge.net/) is an open source remote sensing application and development framework. Opticks supports imagery, motion imagery, SAR, multi-spectral, hyper-spectral, and other types of remote sensing data. Opticks is licensed under LGPL 2.1. This means software developers can easily extend Opticks' functionality and provide that capability to the open source community. It can also be embeded in commercial applications.
  - Opticks also has classified elements, which like OSSIM are modular libraries located on classified client facilities.
  - Opticks is supported by Ball Aerospace, Inc., who insures the security of all source code before military use is authorized.

4.1.2.2.1 How to determine which OSS License for OpenCPI

The best guide on how to pick a license was found in a web posting[12], which very succinctly laid out a yes/no questionnaire on how to pick a license. Below we have answered the questions for converting CPI into OpenCPI; our responses are made in bold and italics.

Do you want to relinquish any control over how your code is used and distributed?

• NO: put it under the BDS/MIT

• YES: Copyright it, and then ask:

Do you want to allow people to use your code in non open-source programs?

–NO: release it under the GPL

–YES: then ask:

If somebody uses your code in their program and sells their program for money, do you want some of that money?

- YES: Dual-license or don't release the source at all and use a closed-source license.

- NO: Use a "commercial-friendly" license, and ask:

If somebody uses your code and improves it (fixes bugs or adds features) do you want to make them give you the improvements back so you can use them too?

–NO: Use a non-reciprocal license.

–YES: Use a reciprocal license

4.1.2.3 OSS License for OpenCPI

The best candidate to release CPI under is the LGPL (GNU LESSER GENERAL PUBLIC LICENSE) Version 3.[13] We have chosen the LGPL for a number reasons, which include:

- The LPGL is a partially closable license: proprietary applications can use unmodified version of the library in a closed source, proprietary licensed product. If changes are made to an LGPL'ed library and distributed, then that modified source code corresponding to that library along with the binary application must be passed along.
- Derivative works must be made available under LGPL.
- The LGPL is primarily used for software libraries, which is what a good portion of what CPI is.
- The LGPL can be mixed with a wide variety of other OSS licenses (see section 4.1.1.5), most specifically the GPL.
- Another more prominent LGPL project that can be compared to OpenCPI is JBoss, which is also a middleware type project.

One of the key features in S3 is to be able to lower the barriers to entry by making sure intellectual property is not captured and locked up by one vendor. We see using the LGPA as a key feature of OpenCPI adoption, which allows an increased competition while speeding developing, which will ultimately lowering costs. The LGPA has some share-and-share-alike features, which are not perceived by industry to be as onerous as the GPL.

OpenCPI value can also be added by supporting 3rd party hardware, including other users' and developers' designs. The idea is to offer OpenCPI on user's preferred hardware platforms. This approach allows a user to develop applications using the CPI interfaces.

4.1.3 OSS Governance

Governance as defined in software source code are the methods and processes necessary to publically release source code. Governance includes answering the issues of how new source code and bug fixes are accepted into the baseline. Also, what is allowed to enter the source code, and what is not, and who in the OpenCPI community is allowed a vote to add new source code into the baseline?

A good governing structure in an OSS software community will ensure that the community will cohesively survive. Governance includes the processes that OpenCPI will use to ensure that code contributions are certified and verified as the 'work of the submitter'.

4.1.3.1 Acceptance of OpenCPI Source Code

To accept new source code, bug fixes, and suggested changes to the baseline OpenCPI code, the OpenCPI community will require a signed form. This will verify that the code is the work of the submitter and that he or she has the right to submit the source and agrees to have their source code released under the OpenCPI OSS license, OpenCPI Source Code Governance Process.

Initially since the OpenCPI community is small, a 'Benevolent Dictator' (BD) model will be used for deciding when things are accepted into the software source codebase. The BD will be a MercFed Senior Embedded System Scientist. As OpenCPI matures, other qualified (and motivated) individuals will be accepted, and the model of acceptance will become a small 'Governance Board'. Everyone in that board will have a vote on when additional codes can be accepted into the code-base. The 'unanimous' model seems to work best for systems such as this where the high reliability of source code is required.

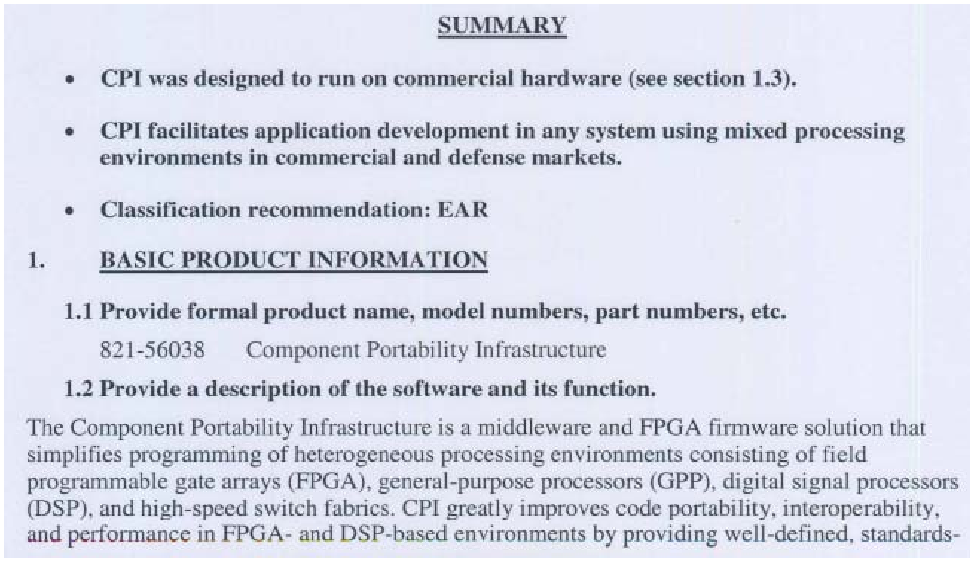
As OpenCPI matures, an experimental release (as an odd number) coupled with a corresponding stable release (as an even number) will be available. The experimental release gives the OpenCPI community a chance to test new features and uncover flaws or bugs before being added into later source code releases.

4.2 International Traffic in Arms Regulations (ITAR)

International Traffic in Arms Regulations (ITAR) governs how and if technology related to the US military and intelligence communities can be shared and exported outside the United States. To determine if a piece of software is an export controlled item, there are two questions that need to be asked. Was that software built for an already ITAR controlled item, and was that software developed for a specific application that is defense related?[2] If the answer to both questions is 'no', then the software isn't export controlled.

Under U.S. law, there are two primary types of export controls. Controls on the export of commercial and "dual use" items (items that are intended for commercial use but can also be applied to military uses) are administered under the Commerce Department's "Export Administration Regulations" (EAR). Controls on "defense articles and defense services and related technical data" (including software) are administered under the State Department.[14]

[2] Consult a lawyer to answer specific questions about ITAR.


SUMMARY

- CPI was designed to run on commercial hardware (see section 1.3).
- CPI facilitates application development in any system using mixed processing environments in commercial and defense markets.
- Classification recommendation: EAR

1. BASIC PRODUCT INFORMATION

1.1 Provide formal product name, model numbers, part numbers, etc.

821-56038 Component Portability Infrastructure

1.2 Provide a description of the software and its function.

The Component Portability Infrastructure is a middleware and FPGA firmware solution that simplifies programming of heterogeneous processing environments consisting of field programmable gate arrays (FPGA), general-purpose processors (GPP), digital signal processors (DSP), and high-speed switch fabrics. CPI greatly improves code portability, interoperability, and performance in FPGA- and DSP-based environments by providing well-defined, standards-

Figure 4: CPI Export Control Summary

CPI was reviewed in 2008 by Mercury Computer Systems and was found not to be an ITAR controlled item and was classified as EAR and submitted to the US Commerce Department for ECCN approval on May 1, 2009 (the response expected back in about 30 days). Figure 4 is a summary of the Mercury report on CPI. A more complete report is located in Appendix A.

4.3 Classified Issues

As discussed previously, OSSIM and Opticks are two well known projects that release source code into the public domain utilizing OSS licenses, and both of these projects have migrated codes into highly classified environments. Both of these projects store public OSS source code on classified networks in code repositories. Additionally, both are stored as modular classified libraries and algorithms, and additional codes can be added depending upon the particular military capabilities that are being developed. This is illustrated for CPI in Figure 5.

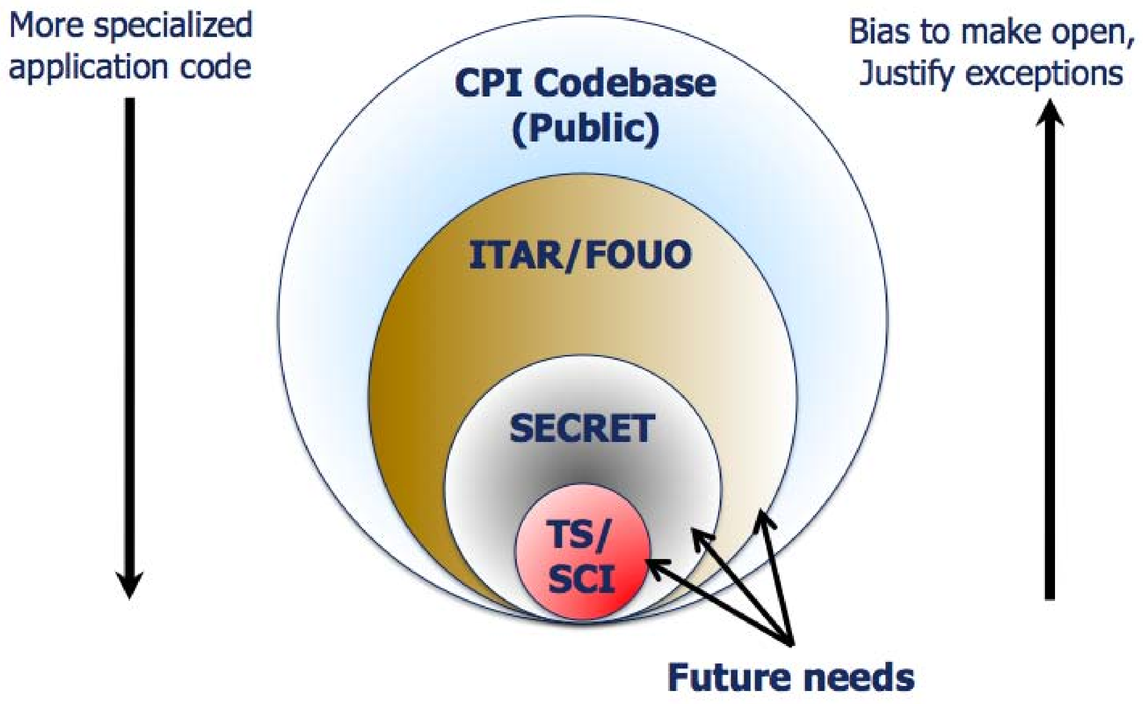

More specialized application code
CPI Codebase (Public)
Bias to make open, Justify exceptions
ITAR/FOUO
SECRET
TS/ SCI
Future needs

Figure 5: OSS and Classified Settings

RadiantBlue and Ball Aerospace, companies who support OSSIM and Opticks respectively, are contracted by the government to keep the software baseline updated with bug fixes, new features, governance and keeping the public and classified versions of the source in sync with each other.

4.4 Best practices of other existing OSS communities

A number of the ideas promulgated previously in this report are culled from the best practices of other successful OSS communities. The primary goal here is a community driven website where the users and developers of OpenCPI can interact and negotiate to add features and mature the code-base. One thing that is required is a community manager to mediate discussions around OpenCPI desired capabilities, new ideas, bug fixes, additional features and find support for problems and issues that occur with CPI. Initially MercFed is the community manager.

An issue that is sometimes overlooked in a code is that a key piece of information is not listed in the design document or the code comments, but might be found in for instance an email or online chat session. The OpenCPI community currently has in place email lists and a wiki to help harvest, store and ensure easy "findability" of unstructured content around OpenCPI capabilities. Existing CPI design documents will be posted inside www.opencpi.org.

4.4.1 Findabilty

MercFed also made the OpenCPI project 'findable' to the outside public world, which means the project was posted to OSS meta-directories like www.freshmeat.net and www.sourceforge.org, and listing/posting the project to mailing lists and sites where real-time developers, users and academics frequent for information. MercFed also needed to optimize how search engines could find OpenCPI and match search terms such as embedded, real-time and open-source-software, so that the www.opencpi.org website would come up at the top of a web search.

4.4.1.1 Logo

One key feature of marketing OpenCPI to help it stand out was its logo, which is presented in Figure 6.

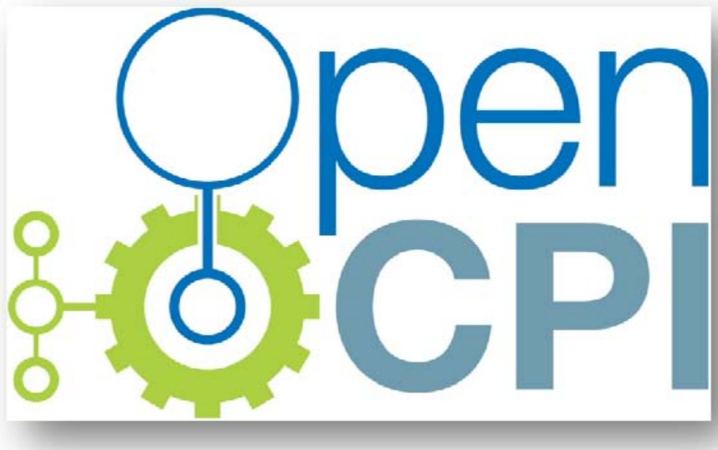


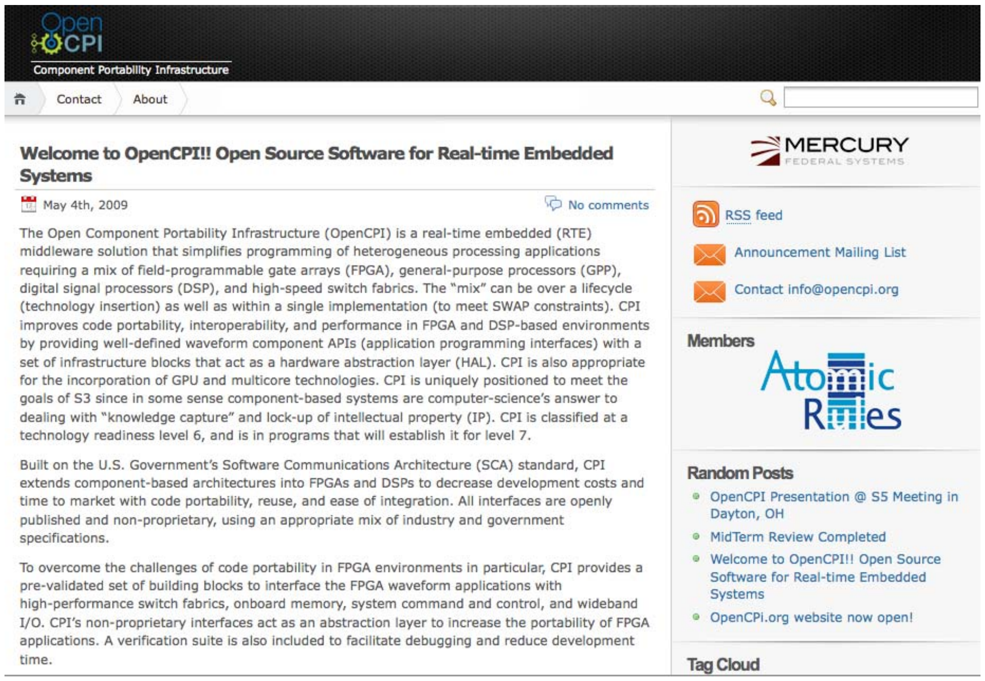
Figure 6: OpenCPI Logo

4.4.2 OpenCPI Community Resources

A crucial part of making OpenCPI a success was to engage the user and developer communities. We made extensive use of various pieces of internet sharing and collaboration, including websites, developer collaboration tools, wiki's, emails and web-based discussion lists to knit together the community.

4.4.2.1 OpenCPI Website

The OpenCPI website is located at www.opencpi.org and is the central place on the web to find information about OpenCPI. There are links to emails list and developer resources here as well. A screen shot of the home page is presented in Figure 7.



Welcome to OpenCPI!! Open Source Software for Real-time Embedded Systems

May 4th, 2009 No comments

The Open Component Portability Infrastructure (OpenCPI) is a real-time embedded (RTE) middleware solution that simplifies programming of heterogeneous processing applications requiring a mix of field-programmable gate arrays (FPGA), general-purpose processors (GPP), digital signal processors (DSP), and high-speed switch fabrics. The "mix" can be over a lifecycle (technology insertion) as well as within a single implementation (to meet SWAP constraints). CPI improves code portability, interoperability, and performance in FPGA and DSP-based environments by providing well-defined waveform component APIs (application programming interfaces) with a set of infrastructure blocks that act as a hardware abstraction layer (HAL). CPI is also appropriate for the incorporation of GPU and multicore technologies. CPI is uniquely positioned to meet the goals of S3 since in some sense component-based systems are computer-science's answer to dealing with "knowledge capture" and lock-up of intellectual property (IP). CPI is classified at a technology readiness level 6, and is in programs that will establish it for level 7.

Built on the U.S. Government's Software Communications Architecture (SCA) standard, CPI extends component-based architectures into FPGAs and DSPs to decrease development costs and time to market with code portability, reuse, and ease of integration. All interfaces are openly published and non-proprietary, using an appropriate mix of industry and government specifications.

To overcome the challenges of code portability in FPGA environments in particular, CPI provides a pre-validated set of building blocks to interface the FPGA waveform applications with high-performance switch fabrics, onboard memory, system command and control, and wideband I/O. CPI's non-proprietary interfaces act as an abstraction layer to increase the portability of FPGA applications. A verification suite is also included to facilitate debugging and reduce development time.

RSS feed
Announcement Mailing List
Contact info@opencpi.org

Members

Atomic Rules

Random Posts

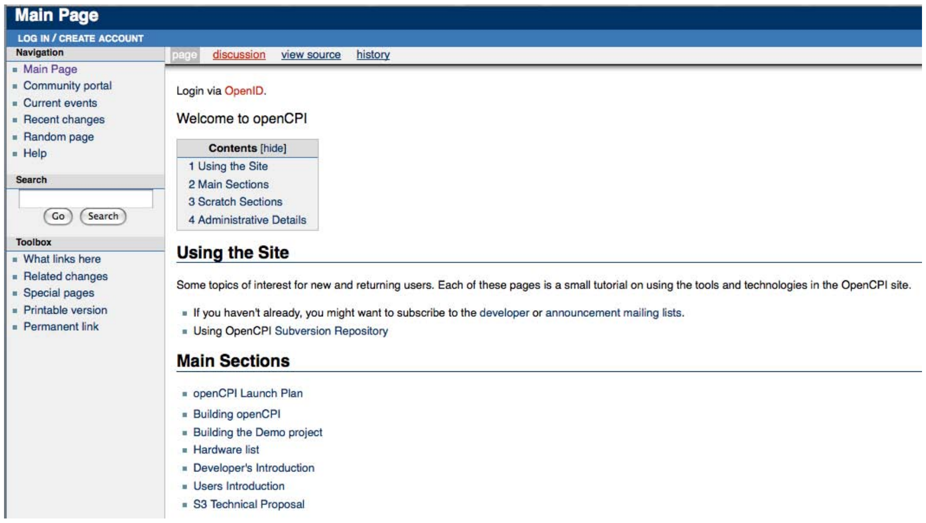
- OpenCPI Presentation @ S5 Meeting in Dayton, OH
- MidTerm Review Completed
- Welcome to OpenCPI!! Open Source Software for Real-time Embedded Systems
- OpenCPI.org website now open!

Tag Cloud

Figure 7: OpenCPI Website

4.4.2.2 Wiki

OpenCPI makes extensive use of the wiki's to simplify the gathering of OpenCPI knowledge located at www.opencpi.org/wiki, as shown in Figure 8.



Login via OpenID.

Welcome to openCPI

Contents [hide]
1 Using the Site
2 Main Sections
3 Scratch Sections
4 Administrative Details

Using the Site

Some topics of interest for new and returning users. Each of these pages is a small tutorial on using the tools and technologies in the OpenCPI site.

- If you haven't already, you might want to subscribe to the developer or announcement mailing lists.
- Using OpenCPI Subversion Repository

Main Sections

- openCPI Launch Plan
- Building openCPI
- Building the Demo project
- Hardware list
- Developer's Introduction
- Users Introduction
- S3 Technical Proposal

Figure 8: Example of OpenCPI Wiki

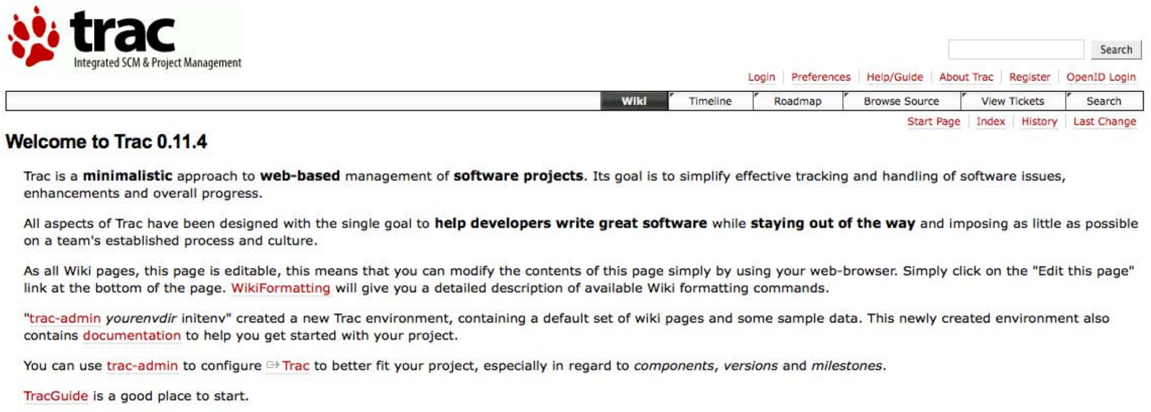
4.4.2.3 Email & Discussion Lists

OpenCPI also has two email lists that individuals call sign up to:

- opencpi_announce@lists.opencpi.org for general OpenCPI announcements; and
- opencpi_dev-request@lists.opencpi.org for OpenCPI software developers.

4.4.2.4 Software Source Code & OpenCPI Roadmap

OpenCPI deployed its code to the website: http://trac.opencpi.org, a screen shot of which is shown in Figure 9. Trac is a software road-mapping tool that MercFed will use to track bugs and desired new features.



Welcome to Trac 0.11.4

Trac is a **minimalistic** approach to **web-based** management of **software projects**. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.

All aspects of Trac have been designed with the single goal to **help developers write great software** while **staying out of the way** and imposing as little as possible on a team's established process and culture.

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. WikiFormatting will give you a detailed description of available Wiki formatting commands.

"trac-admin *yourenvdir* initenv" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains documentation to help you get started with your project.

You can use trac-admin to configure Trac to better fit your project, especially in regard to *components*, *versions* and *milestones*.

TracGuide is a good place to start.

Figure 9: OpenCPI Software Source Code Website

5.0 CONCLUSIONS

Building a community around an open source software project is complex. Our concept here was to gather like-minded individuals who have the same problem sets around real-time and embedded software. In the future we will need to broaden the community.

5.1 OpenCPI Business Model

One of the best ways to grow and maintain a healthy software baseline is to ensure that there are revenue opportunities available. The most successful company that has been able to monetize open source software is RedHat with its two main product lines RedHat Linux and JBoss. RedHat provides a subscription service to both of these products that include help line support, rapid updates of bug and new features. Additionally, tool sets are provided that help manage the product lines internal to a customer. Sun Microsystems also sells a subscription support service for all of its associated open source software products including OpenSolaris and Java.

There are a number of different business models[15], which are in order of lower to higher value:

1. Selling installation, service and support with the software.
2. Versioning the software, with the free version as an entry-level offering and other, more advanced versions as value-added offerings.
3. Integrating the software with other parts of the customer's information technology infrastructure.
4. Providing proprietary complements to open source software.

MercFed anticipates selling subscription services of updates of OpenCPI source code coupled with proprietary complements to OpenCPI. An unanswered question is what it is that a subscription should be linked to. Redhat links their subscriptions to auditable items, such as a personal computers or servers. MercFed needs to determine what the OpenCPI link should be. Possibilities include a computer chip, a board, an entire computer, a system, a military program or an enterprise. This decision will be made in conjunction with the development of the OpenCPI value proposition.

5.1.1 Other Deliverables

There are additionally three other deliverables as part of this research project:

- Open Source Software (OSS) Roadmap for Defense Industry, which presented how a company can convert previously closed source software into open source, and is a subset of this report.
- OpenCPI.org Administration Guide, which details how to manage the OpenCPI website.
- OpenCPI.org Users Guide, which describes how a user can navigate and use the OpenCPI website.

5.2 Lower Barriers to Entry

To broaden the community, the barrier to entry to using OpenCPI needs to be lowered. This means simplifying the installation and use process of OpenCPI. A hardware bill of materials, presented in Appendix B, is available for download at OpenCPI.org. This is a simplified description of what an example OpenCPI system. Our developer and build notes about the sample systems we are also building are also being posted there.

Methods to lower barriers to entry to use OpenCPI include:

- Hardware: At http://opencpi.org/wiki current and new ideas about the types and configurations of hardware deployed with OpenCPI can be posted for others to use.
- Software: OpenCPI progressively evolve and become easier to download and use quickly ‘out of the box.’ This includes building better software developer kits (SDKs) and more extensive installation notes.
- Communication channels: OpenCPI needs to have multiple communication channels, including a blog, wiki, email lists, twitter feed and eventually an always on IRC (internet chat relay) chat channel so that developers and users feel they always have 24/7 distributed support.

5.3 Engage the Academic & Research Community

Most open source software projects have seen great adoption by users in the academic and research communities since the technologies are so accessible to use and extend. OpenCPI has reached out to the academic community and is slowly gaining interest and traction. A key feature for future work would be to fund academic research and development projects to show that OpenCPI can be a key technology enabler.

5.4 Engage Defense & Industry

The community that will have the most to gain from OpenCPI across a number of application areas is the military, so it is vital that MercFed engages current and potential military users of OpenCPI. Currently OpenCPI is a part of two military science and technology programs:

- Joint Counter RCIED Electronic Warfare (JCREW) CIED Defeat Experimental Software Platform. RACID is an acronym for Radio Controlled Improvised Explosive Device.
- United States Air Force (USAF) High Data Rate-Radio Frequency (HDR-RF) SATCOM Modem Program, Phase 1.

Both programs are ongoing and have significant government and contractor interest in the OpenCPI software product line. Currently the following defense partners are evaluating the value in OpenCPI, and are considering new and future programs for inclusion of OpenCPI:

- Aurora Flight Systems, Inc. (Manassas, VA).
- General Dynamics, Inc., Advanced Information Systems (Reston, VA).
- BAE America, Inc. (Herndon, VA).

Additionally, there are partners outside the traditional defense sphere whose products are extensively used in the defense industry have expressed interest include:

- Advanced Micro Devices (AMD), Inc. (computer chip vendor)
- Xilinx, Inc. (FPGA vendor)
- Achronix, Inc. (FPGA vendor)

5.5 Add Features and Applications

As MercFed endeavors to build the OpenCPI ecosystem, new features will be required and added to OpenCPI. Users of OpenCPI will be encouraged to acknowledge what applications have been built on top of OpenCPI.

6.0 REFERENCES

1. Technology Readiness Level Definition: http://en.wikipedia.org/wiki/Technology_Readiness_Levels.
2. Open Source Initiative OSS license list: www.opensource.org/licenses/alphabetical.
3. OSI, Open source definition: www.opensource.org/docs/osd.
4. OSI website: www.opensource.org.
5. Rosen, Lawrence Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, Pretence Hall PTR, Upper Saddle River, NJ, 2005, p. 22.
6. *Ibid*, Rosen, p. 23
7. Wheeler, David: http://www.dwheeler.com/essays/floss-license-slide.html.
8. *Ibid*, D. Wheeler.
9. http://java.net/choose_license.csp.
10. Free Software Foundation License Compliance: www.fsf.org/licensing/compliance.
11. *Use of Free and Open Source Software in the U.S. Department of Defense*, V. 1.2.02, released November 6, 2002.
12. Burnette, Ed, ZDNet.com Blog, June 14th, 2006, HOWTO: Pick an open source license: http://blogs.zdnet.com/Burnette/?p=130.
13. GPL OSS License: www.gnu.org/licenses/lgpl.html.
14. Wheeler, David: http://www.dwheeler.com/essays/dod-oss-qa.html#itar.
15. Chesbrough, Henry William, Open Business Models: How to thrive in the New Innovation Landscape, Harvard Business School Press, Boston, MA, 2006, p. 45.

7.0 LIST OF ABBRVIATIONS AND ACRONYMS

API	Application Programming Interface
BD	Benevolent Dictator
BSD	Berkeley Software Distribution
CIED	Counter Improvised Explosive Devices
CORBA	Common Object Request Broker Architecture
COTS	Commercial-Off-The-Shelf
CPI	Component Portability Infrastructure
DoD	Department of Defense
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EAR	Export Administration Regulations
FLOSS	Free/ Libre/ Open Source Software
FIFO	First In, First Out
FOUO	FOR OFFICIAL USE ONLY
FPGA	Field-Programmable Gate Array
GigE	Gigabit Ethernet
GNU	Recursive acronym for “GNU’s not Unix!”
GPGPU	General-Purpose computing on Graphics Processing Unit
GPL	General Purpose License
GPP	General-Purpose Processors
GPU	Graphics Processing Unit
HAL	Hardware Abstraction Layer
HDR-RF	High Data Rate-Radio Frequency
IED	Improvised Explosive Device
I/O	Input/Output
IP	Intellectual Property
ISR	Intelligence, Surveillance and Reconnaissance
ITAR	International Traffic in Arms Regulations
JBoss	A division of Red Hat
JCREW	Joint Counter RCIED Electronic Warfare
LGPL	Lesser General Public License
MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
NT	Intelligence

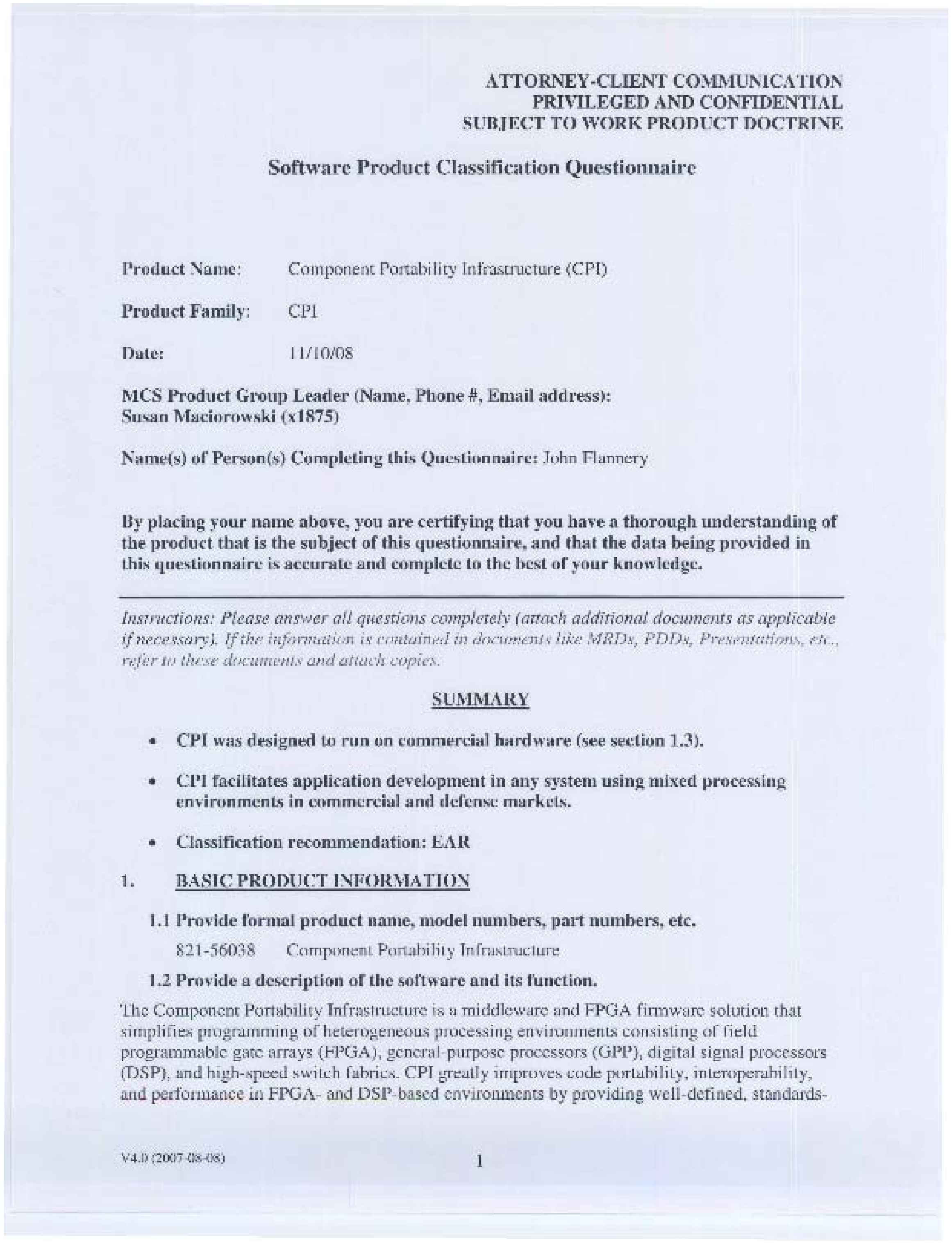
OCP	Open Core Protocol
OpenCPI	Open Component Portability Infrastructure
OSI	Open Source Initiative
OSS	Open Source Software
OSSIM	Open Source Software Image Map
OST	Open Source Initiative
POSIX	Portable Operating System Interface [for Uni**x]**
RACID	Radio Controlled Improvised Explosive Device
RTE	Real-Time Embedded
RTOS	Real-Time Operating System
S3	Software Systems Stockroom
SA Forum	Service Availability Forum™
SATCOM	Satellite Communications
SCA	Software Communications Architecture
SDK	Software Developer Kit
SDR	Software Defined Radio
SIGINT	Signals Intelligence
SRAM	Static Random Access Memory
TRL	Technology Readiness Level
VSIPL	Vector Signal Image Processing Library
xxxINT	Signals, Communications, Electronics, etc. - Intelligence

APPENDIX A: ADDITIONAL REFERENCES

- Open Source Licensing, Lawrence Rosen, Pretence Hall 2005.
- Open Source & Free Software Licensing, St. Laurent, Andrew M., O'Reilly Media 2004.
- Open Source 2.0, DiBona, Cooper & Stone (Editors), O'Reilly Media 2006.
- Open Source Initiative, http://www.opensource.org/.
- Intellectual Property and Open Source, Van Lindberg, O'Reilly Media 2008.
- Weber, Steven, The Success of Open Source, Harvard University Press, 2004.
- US Navy Open Architecture Guidance - https://acc.dau.mil/CommunityBrowser.aspx?id=105662.
- AS&C OTD Roadmap - www.acq.osd.mil/actd/articles/OTDRoadmapFinal.pdf.
- OTD Website – www.opentechdev.org.
- Clinger-Cohen Act of 1996 (in P.L. 104-106), 5123, Performance and Results-Based Management.
- Title 5, United States Code, 306, Strategic Plans (part of Government Performance and Results Act (GPRA)).
- CJCSI 3170.01, "Requirements Generation System (Formerly MOP 77)," 6/13/97.
- Title 10, United States Code, Section 2377, Preference for acquisition of commercial items.
- Federal Acquisition Regulation, Part 6.3, Other Than Full and Open Competition.
- Defense Federal Acquisition Regulation Supplement, Appendix D, Component Breakout.
- Clinger-Cohen Act of 1996 (in P.L. 104-106), 5122, Capital Planning And Investment Control, 5123, Performance And Results-Based Management, and 5202, Incremental Acquisition Of Information Technology.
- ASD(C3I) memorandum, "Use of the Ada Programming Language," April 29, 1997.
- Department of Defense Directive 4630.5, Compatibility, Interoperability, and Integration of Command, Control, Communications, and Intelligence (C3I) Systems, Nov.12, 1992.
- Department of Defense Instruction 4630.8, Procedures for Compatibility, Interoperability, and Integration of Command, Control, Communications, and Intelligence (C3I) Systems, November 18, 1992.
- CJCS Instruction 6212.01A, Compatibility, Interoperability, and Integration of Command, Control, Communications, Computers, and Intelligence Systems, June 30, 1995.
- Clinger-Cohen Act of 1996 (in P.L. 104-106), 5202, Incremental Acquisition Of Information Technology.
- Title 44, United States Code, 3506, Federal agency responsibilities (amended by Public Law 104-13, Paperwork Reduction Act (PRA) of 1995).
- Memo: CIO John P. Stenbit SUBJECT: Open Source Software (OSS) in the Department of Defense (DoD), May 28, 2003.

- MITRE Corporation Report: Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense, Version 1.2.04, January 2, 2003, Report # MP 02 W0000101.
- IBM VC calls for 'open' hardware, Richard Goering, EE Times, 04/08/2005, www.eetimes.com/news/design/showArticle.jhtml?articleID=160502705.
- Raymond, E.S., The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly Publishers, 2001.

Appendix B: Mercury CPI ITAR Report



ATTORNEY-CLIENT COMMUNICATION
PRIVILEGED AND CONFIDENTIAL
SUBJECT TO WORK PRODUCT DOCTRINE

Software Product Classification Questionnaire

Product Name:	Component Portability Infrastructure (CPI)
Product Family:	CPI
Date:	11/10/08

MCS Product Group Leader (Name, Phone #, Email address):
Susan Maciorowski (x1875)

Name(s) of Person(s) Completing this Questionnaire: John Flannery

By placing your name above, you are certifying that you have a thorough understanding of the product that is the subject of this questionnaire, and that the data being provided in this questionnaire is accurate and complete to the best of your knowledge.

Instructions: Please answer all questions completely (attach additional documents as applicable if necessary). If the information is contained in documents like MRDs, PDDs, Presentations, etc., refer to these documents and attach copies.

SUMMARY

- **CPI was designed to run on commercial hardware (see section 1.3).**
- **CPI facilitates application development in any system using mixed processing environments in commercial and defense markets.**
- **Classification recommendation: EAR**

1. **BASIC PRODUCT INFORMATION**

1.1 Provide formal product name, model numbers, part numbers, etc.

821-56038 Component Portability Infrastructure

1.2 Provide a description of the software and its function.

The Component Portability Infrastructure is a middleware and FPGA firmware solution that simplifies programming of heterogeneous processing environments consisting of field programmable gate arrays (FPGA), general-purpose processors (GPP), digital signal processors (DSP), and high-speed switch fabrics. CPI greatly improves code portability, interoperability, and performance in FPGA- and DSP-based environments by providing well-defined, standards-

V4.0 (2007-08-08) 1

with the industry structure and value chain shown in Figure 1, consists of multiple segments, including:

- Commercial wireless LAN;
- Military;
- Public safety;
- Cellular handsets;
- Automotive;
- Test equipment;
- Transportation; and
- Consumer electronics.

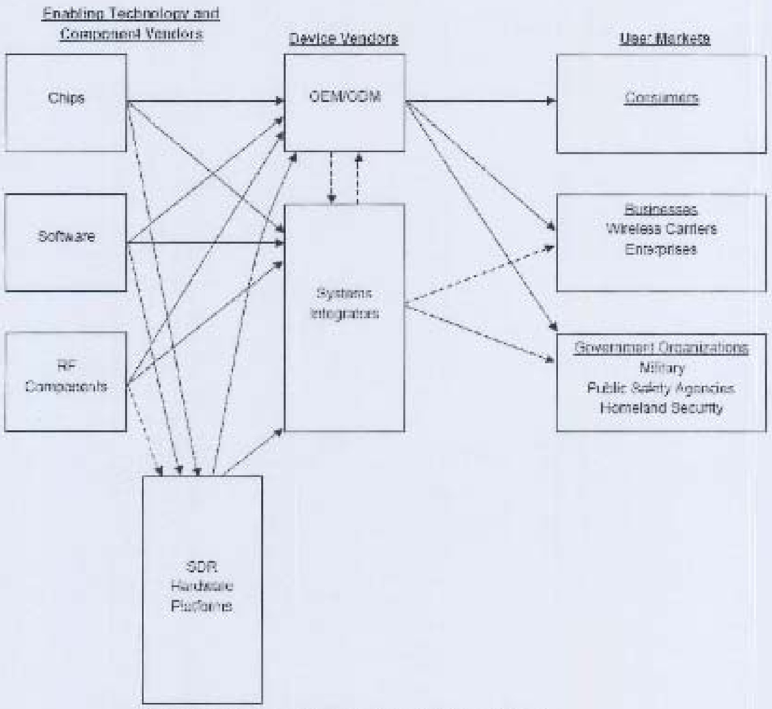
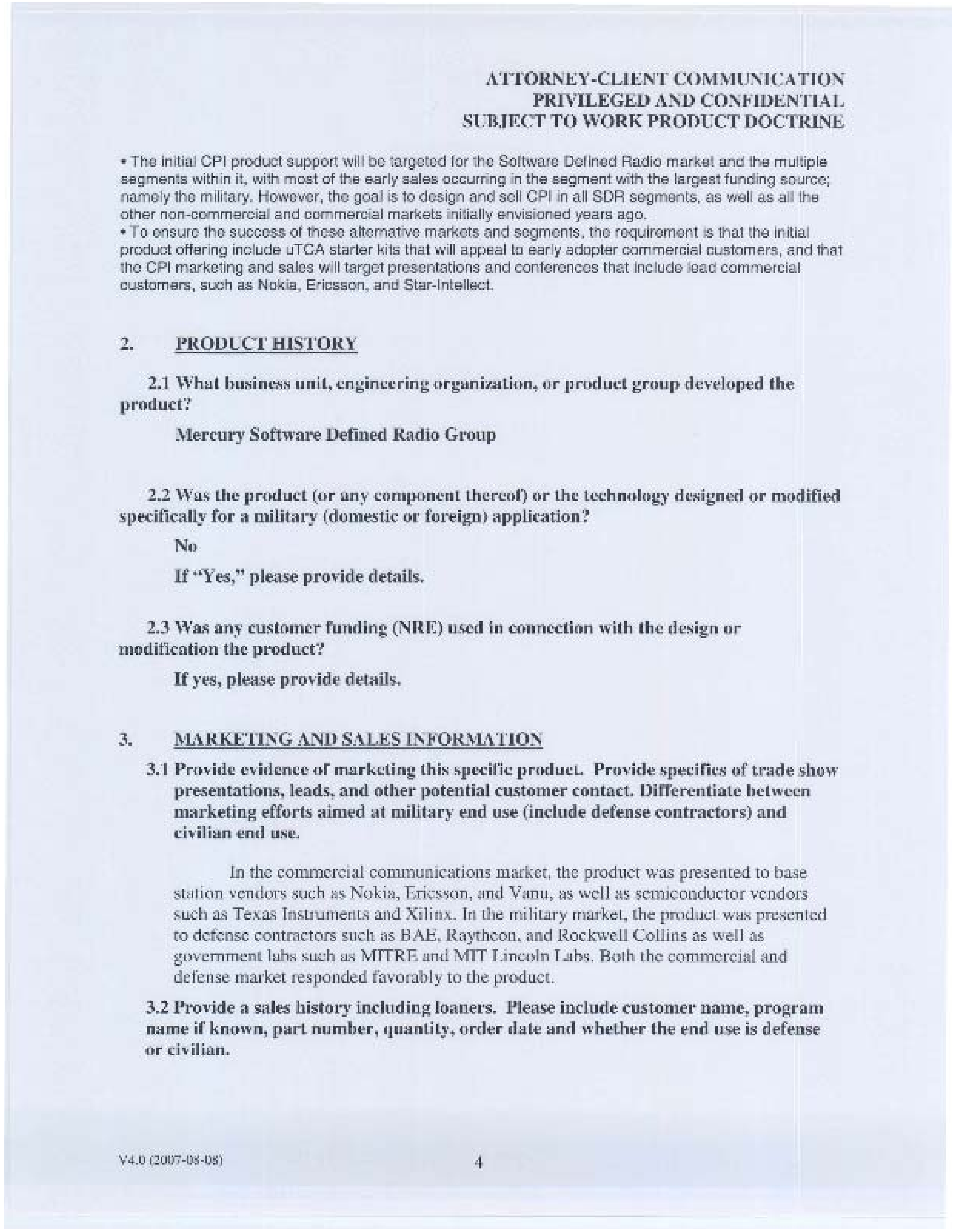


Figure 1: SDR Industry Structure and Value Chain

All of these market segments offered significant new opportunities for the MedVIP concept/components. Therefore, as a result of this emerging SDR market, internally we redirected our focus to a solution that could support multiple markets and segments, and renamed the key middleware and component elements to Stream Co-Processing Engine (SCE). Eventually, Mercury renamed SCE to Component Portability Infrastructure (CPI) as an external marketing name for the product concept.

Key expectations and marketing requirements for CPI may be summarized as follows:



ATTORNEY-CLIENT COMMUNICATION
PRIVILEGED AND CONFIDENTIAL
SUBJECT TO WORK PRODUCT DOCTRINE

• The initial CPI product support will be targeted for the Software Defined Radio market and the multiple segments within it, with most of the early sales occurring in the segment with the largest funding source; namely the military. However, the goal is to design and sell CPI in all SDR segments, as well as all the other non-commercial and commercial markets initially envisioned years ago.
• To ensure the success of these alternative markets and segments, the requirement is that the initial product offering include uTCA starter kits that will appeal to early adopter commercial customers, and that the CPI marketing and sales will target presentations and conferences that include lead commercial customers, such as Nokia, Ericsson, and Star-Intellect.

2. PRODUCT HISTORY

2.1 What business unit, engineering organization, or product group developed the product?

Mercury Software Defined Radio Group

2.2 Was the product (or any component thereof) or the technology designed or modified specifically for a military (domestic or foreign) application?

No

If "Yes," please provide details.

2.3 Was any customer funding (NRE) used in connection with the design or modification the product?

If yes, please provide details.

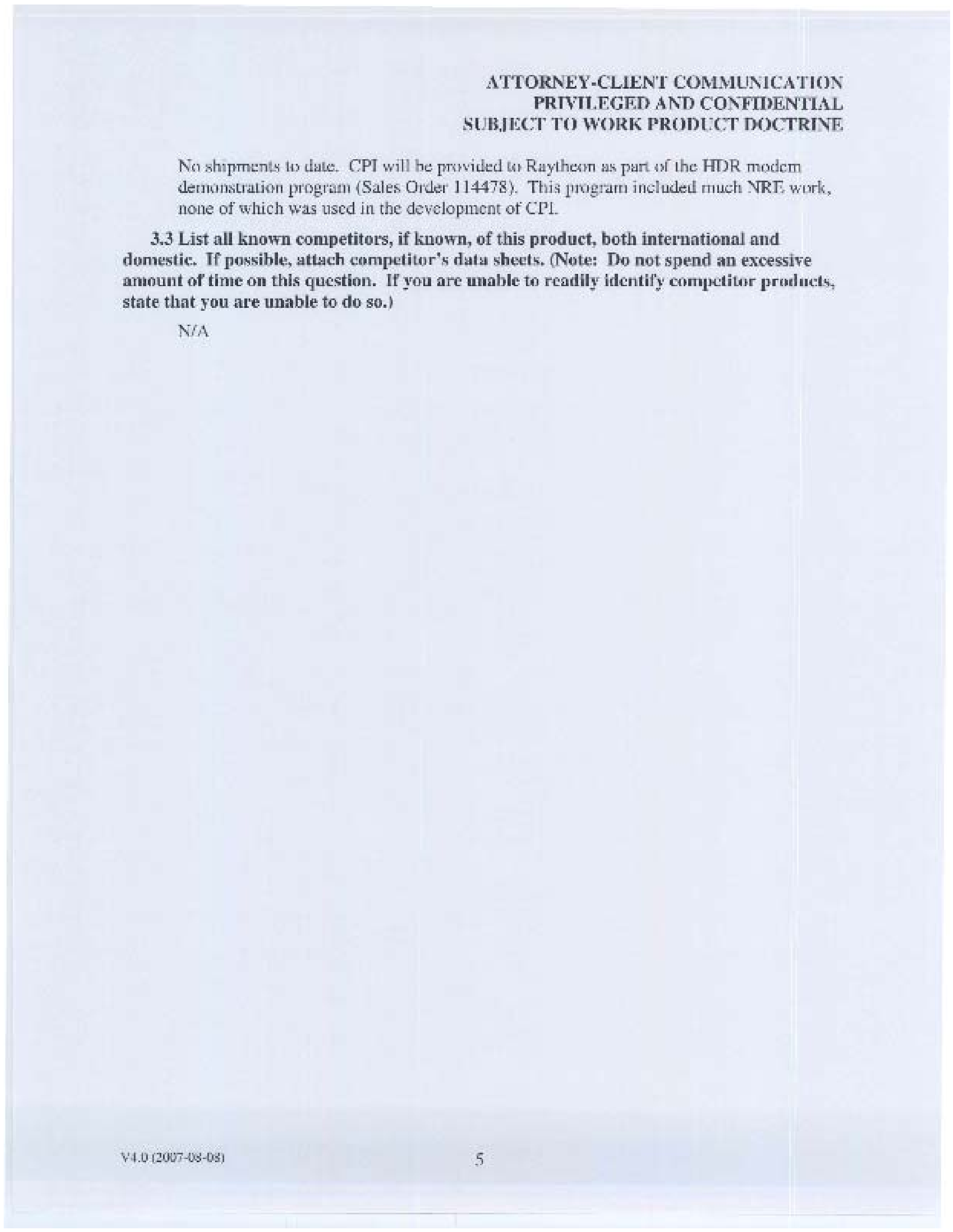
3. MARKETING AND SALES INFORMATION

3.1 Provide evidence of marketing this specific product. Provide specifics of trade show presentations, leads, and other potential customer contact. Differentiate between marketing efforts aimed at military end use (include defense contractors) and civilian end use.

In the commercial communications market, the product was presented to base station vendors such as Nokia, Ericsson, and Vanu, as well as semiconductor vendors such as Texas Instruments and Xilinx. In the military market, the product was presented to defense contractors such as BAE, Raytheon, and Rockwell Collins as well as government labs such as MITRE and MIT Lincoln Labs. Both the commercial and defense market responded favorably to the product.

3.2 Provide a sales history including loaners. Please include customer name, program name if known, part number, quantity, order date and whether the end use is defense or civilian.

V4.0 (2007-08-08) 4



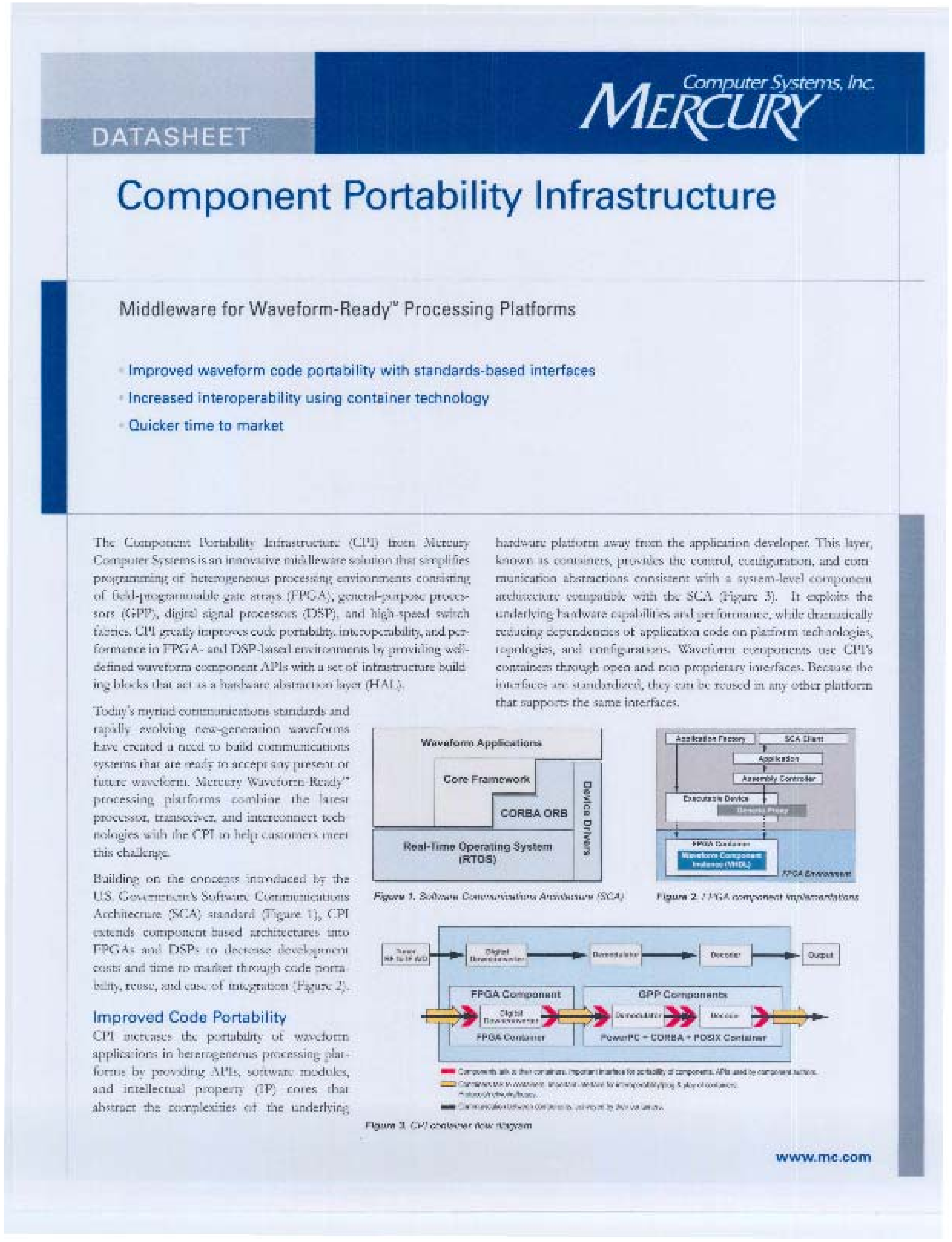
ATTORNEY-CLIENT COMMUNICATION
PRIVILEGED AND CONFIDENTIAL
SUBJECT TO WORK PRODUCT DOCTRINE

No shipments to date. CPI will be provided to Raytheon as part of the HDR modem demonstration program (Sales Order 114478). This program included much NRE work, none of which was used in the development of CPI.

3.3 List all known competitors, if known, of this product, both international and domestic. If possible, attach competitor's data sheets. (Note: Do not spend an excessive amount of time on this question. If you are unable to readily identify competitor products, state that you are unable to do so.)

N/A

APPENDIX C: CPI DATASHEET



DATASHEET

Computer Systems, Inc. MERCURY

Component Portability Infrastructure

Middleware for Waveform-Ready™ Processing Platforms

- Improved waveform code portability with standards-based interfaces
- Increased interoperability using container technology
- Quicker time to market

The Component Portability Infrastructure (CPI) from Mercury Computer Systems is an innovative middleware solution that simplifies programming of heterogeneous processing environments consisting of field-programmable gate arrays (FPGA), general-purpose processors (GPP), digital signal processors (DSP), and high-speed switch fabrics. CPI greatly improves code portability, interoperability, and performance in FPGA- and DSP-based environments by providing well-defined waveform component APIs with a set of infrastructure building blocks that act as a hardware abstraction layer (HAL).

Today's myriad communications standards and rapidly evolving new-generation waveforms have created a need to build communications systems that are ready to accept any present or future waveform. Mercury Waveform-Ready™ processing platforms combine the latest processor, transceiver, and interconnect technologies with the CPI to help customers meet this challenge.

Building on the concepts introduced by the U.S. Government's Software Communications Architecture (SCA) standard (Figure 1), CPI extends component based architectures into FPGAs and DSPs to decrease development costs and time to market through code portability, reuse, and ease of integration (Figure 2).

Improved Code Portability

CPI increases the portability of waveform applications in heterogeneous processing platforms by providing APIs, software modules, and intellectual property (IP) cores that abstract the complexities of the underlying hardware platform away from the application developer. This layer, known as containers, provides the control, configuration, and communication abstractions consistent with a system-level component architecture compatible with the SCA (Figure 3). It exploits the underlying hardware capabilities and performance, while dramatically reducing dependencies of application code on platform technologies, topologies, and configurations. Waveform components use CPI's containers through open and non proprietary interfaces. Because the interfaces are standardized, they can be reused in any other platform that supports the same interfaces.

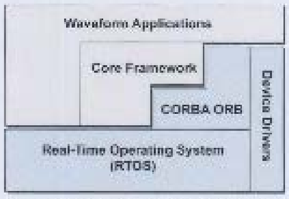


Figure 1. Software Communications Architecture (SCA)

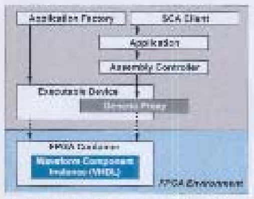


Figure 2. FPGA component implementations

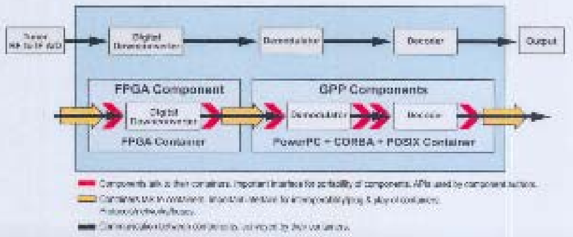
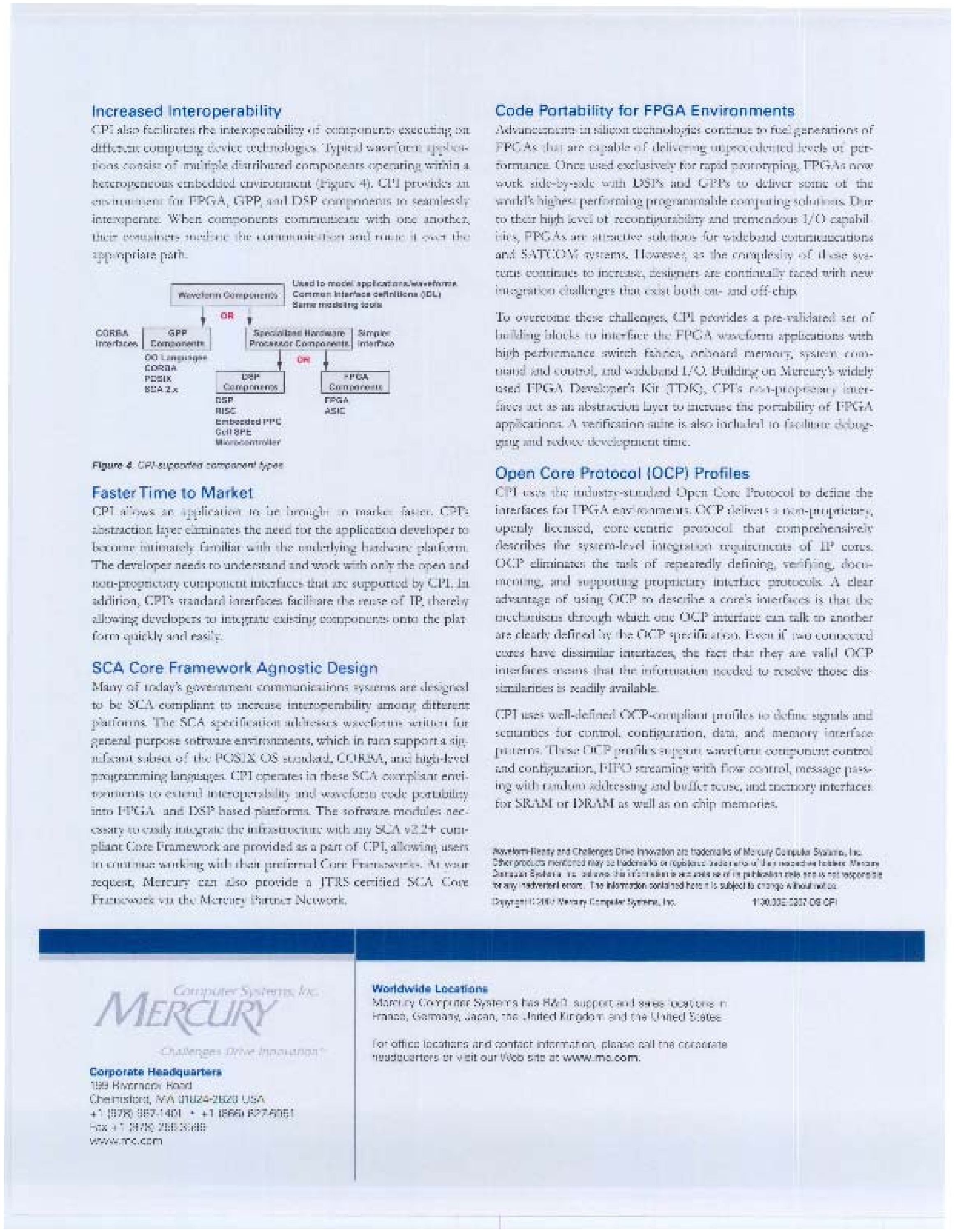


Figure 3. CPI container flow diagram

www.mc.com



Increased Interoperability

CPI also facilitates the interoperability of components executing on different computing device technologies. Typical waveform applications consist of multiple distributed components operating within a heterogeneous embedded environment (Figure 4). CPI provides an environment for FPGA, GPP, and DSP components to seamlessly interoperate. When components communicate with one another, their containers mediate the communication and route it over the appropriate path.

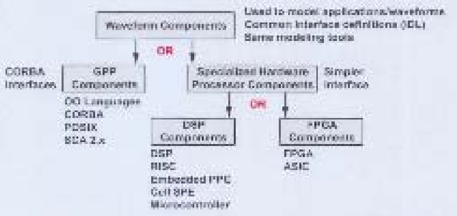


Figure 4. CPI-supported component types

Faster Time to Market

CPI allows an application to be brought to market faster. CPI's abstraction layer eliminates the need for the application developer to become intimately familiar with the underlying hardware platform. The developer needs to understand and work with only the open and non-proprietary component interfaces that are supported by CPI. In addition, CPI's standard interfaces facilitate the reuse of IP, thereby allowing developers to integrate existing components onto the platform quickly and easily.

SCA Core Framework Agnostic Design

Many of today's government communications systems are designed to be SCA compliant to increase interoperability among different platforms. The SCA specification addresses waveforms written for general purpose software environments, which in turn support a significant subset of the POSIX OS standard, CORBA, and high-level programming languages. CPI operates in these SCA compliant environments to extend interoperability and waveform code portability into FPGA and DSP based platforms. The software modules necessary to easily integrate the infrastructure with any SCA v2.2+ compliant Core Framework are provided as a part of CPI, allowing users to continue working with their preferred Core Frameworks. At your request, Mercury can also provide a JTRS certified SCA Core Framework via the Mercury Partner Network.

Code Portability for FPGA Environments

Advancements in silicon technologies continue to fuel generations of FPGAs that are capable of delivering unprecedented levels of performance. Once used exclusively for rapid prototyping, FPGAs now work side-by-side with DSPs and GPPs to deliver some of the world's highest performing programmable computing solutions. Due to their high level of reconfigurability and tremendous I/O capabilities, FPGAs are attractive solutions for wideband communications and SATCOM systems. However, as the complexity of these systems continues to increase, designers are continually faced with new integration challenges that exist both on- and off-chip.

To overcome these challenges, CPI provides a pre-validated set of building blocks to interface the FPGA waveform applications with high performance switch fabrics, onboard memory, system command and control, and wideband I/O. Building on Mercury's widely used FPGA Developer's Kit (FDK), CPI's non-proprietary interfaces act as an abstraction layer to increase the portability of FPGA applications. A verification suite is also included to facilitate debugging and reduce development time.

Open Core Protocol (OCP) Profiles

CPI uses the industry-standard Open Core Protocol to define the interfaces for FPGA environments. OCP delivers a non-proprietary, openly licensed, core-centric protocol that comprehensively describes the system-level integration requirements of IP cores. OCP eliminates the task of repeatedly defining, verifying, documenting, and supporting proprietary interface protocols. A clear advantage of using OCP to describe a core's interfaces is that the mechanisms through which one OCP interface can talk to another are clearly defined by the OCP specification. Even if two connected cores have dissimilar interfaces, the fact that they are valid OCP interfaces means that the information needed to resolve those dissimilarities is readily available.

CPI uses well-defined OCP-compliant profiles to define signals and semantics for control, configuration, data, and memory interface patterns. These OCP profiles support waveform component control and configuration, FIFO streaming with flow control, message passing with random addressing and buffer reuse, and memory interfaces for SRAM or DRAM as well as on chip memories.

Computer Systems, Inc.
MERCURY
Challenges Drive Innovation™

Corporate Headquarters
199 Riverneck Road
Chelmsford, MA 01824-2820 USA
+1 (978) 967-1401 • +1 (866) 627-6951
Fax +1 (978) 256-3599
www.mc.com

Worldwide Locations
Mercury Computer Systems has R&D, support and sales locations in France, Germany, Japan, the United Kingdom and the United States.

For office locations and contact information, please call the corporate headquarters or visit our Web site at www.mc.com.

APPENDIX D: OPENCPI HARDWARE BILL OF MATERIALS

OpenCPI Reference Platform, COTS Component Specification

Shepard Siegel, Atomic Rules LLC (Shepard.Siegel@atomicrules.com)

This document specifies the COTS components used for the OpenCPI FPGA Reference Platform (OC-FRP). The parts and vendors specified herein have been selected for their ubiquity and low-cost. For most of the items listed, component substitutions may be made. However, the burden of testing is then upon the user. Pricing is non-binding, quantity-one, and guidance only.

Development and Target Computer, Hardware Components

It is possible to use one computer as both a development machine and a target. Table 1 lists components central to the performance of this system.

Table D1: FPGA Reference Platform, Hardware Components

Line	#	Manufacturer	Part Number	Cost	Description
1	1	www.evga.com	132-BL-E758-A1	$300.	X58 Motherboard
2	1	www.intel.com	Corei7 920	$280.	Nehalem 2.66GHz x4 Processor
3	2	www.corsair.com	DDR3 3x2GB	$190.	6x2GB (12GB) DDR3 1600 Memory
4	1	www.evga.com	01G-P3-1280	$335.	GTX280 240-core GPU
5	1	www.enermax.com	EMD625AWT	$160.	625W MODU82+ Power Supply
6	1	www.westerndigital.com	WD10EADS	$105	1TB SATA Disk Drive
7	1	www.antec.com	P182	$120.	ATX Mid Tower Computer Case
8	1	www.xilinx.com	HW-V5-ML555-G	$2200.	Xilinx ML555 V5 Dev Kit

Notes:

- Cost estimated, quantity one, as observed online Q1-2009

All components in the table above, except for the ML555, are available from computer component retailers including www.newegg.com. The rationale for the selection of line items 1-7 was to satisfy the requirement of identifying one known and stable "x58-based Corei7 PC with 12GB RAM, strong GPU and reliable 625W power supply". Deviating from these specific selections should not significantly alter the system behavior. For example, replacing the GTX280 GPU (cutting-edge "strong" in Q1-2009) with a different GPU should not change the behavior of OpenCPI applications.

The Xilinx ML555 Development Kit is available from www.avnet.com and www.nuhorizons.com . This development kit includes the "Platform USB /JTAG programming cable". Not included with this kit are the optional 1000BASE-T SFP Transceivers (Finisar FCMJ-8521-3) from Avnet for $80. One or two are required for direct-to-FPGA GBE connectivity.

Additionally, not included in this table were components which may vary based on the user's specific requirements. These include the CPU heat sink assembly, CD/DVD drive, keyboard, mouse, monitor and peripheral hardware that will be needed to complete the system.

An open-frame computer case may be used as an alternative to the enclosed case specified in line item 7 when frequent access to board hardware is desired, or where there may be mechanical interference from other PCIe add-in cards with a taller profile. For example, PCIe cards with top-mounted FMC/VITA-57 connectors, such as the Xilinx ML605, will protrude in the vertical dimension such that the case side panel may not be fitted. One such open-frame case alternative offering is the “HSPC Top Deck Tech Station (standard size)” from www.highspeedpc.com for about $90.

Development and Target Computer, Software Components

Note: The information below is the latest available pricing, but can be expected to change prior to document release. Xilinx ISE 11.1 is presently in Beta L.31. It is anticipated that Beta L.33 or greater will be available for the production release.

Table D2: FPGA Reference Platform, Software Components

Line	Manufacturer	Part Number	Cost	Description
1	www.ubuntu.com	8.10-desktop-amd64	Free	Ubuntu 8.10 “Intrepid Ibex” 64-bit
2	www.xilinx.com	TBD	$2995.	Xilinx ISE 11.1 Logic Edition, Node-Locked